UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería informática

TRABAJO FIN DE GRADO

Gamificación aplicada al aprendizaje del diseño orientado a objetos

Javier Muñoz Tabuenca
Tutor: Esther Guerra Sánchez

Julio 2015

Resumen

El uso del juego como método de aprendizaje para motivar a los alumnos a adquirir conocimientos y competencias de forma más amena es una técnica utilizada desde hace tiempo. Pero no fue hasta principios de este milenio cuando fue bautizado con el término gamificación y comenzó a cobrar importancia.

Este método cada vez es más usado y no sólo con niños, también se han aplicado estas técnicas al ámbito profesional, desde recreaciones de quirófano para médicos hasta simuladores de vuelo para pilotos.

El objetivo de este proyecto es crear una plataforma que permita a los alumnos mejorar sus conocimientos sobre el diseño orientado a objetos, más concretamente sobre los diagramas de clases UML, utilizando técnicas de gamificación.

Con este objetivo, se ha desarrollado Diagame, un sistema que consta de un módulo de gestión, un módulo de generación de ejercicios, y un módulo de juego. El módulo de gestión permite a los profesores crear enunciados y sus diagramas de clases solución. El módulo de generación de ejercicios crea versiones erróneas de las soluciones proporcionadas por el profesor mediante la inyección aleatoria de distintos tipos errores, con objeto de obtener automáticamente ejercicios que ofrecer a los estudiantes.

Por último, el módulo de juego, donde el alumno adoptará el papel de dueño de una pequeña empresa. Para incrementar el valor de la empresa el usuario deberá resolver correctamente diagramas de clases UML que le aportarán unos beneficios. Con este dinero el alumno puede contratar empleados y bonificadores especiales que incrementen el valor de los próximos diagramas que resuelva el usuario, para obtener beneficios muchos mayores y subir posiciones en el ranking de empresas más valiosas.

Palabras clave

Gamificación, Orientación a Objetos, Diagramas de Clases UML, Diseño de Software

Abstract

The use of games as a learning method to motivate students to achieve knowledge and skills in a more pleasant way is a technique that has long been used. But it was not until early this millennium when the term gamification was coined and began to gain importance.

This method is being increasingly used not only with children, but it has also been applied in professional contexts, ranging from surgical virtualizations for doctors to flight simulators for pilots.

The goal of this project is to create a platform that allows students to improve their knowledge on object-oriented design, more specifically on UML class diagrams, by means of gamification techniques.

For this purpose, we have created Diagame, a system that includes a management module, an exercise generation module, and a game module. The management module allows teachers to create problem statements and their solution diagrams. The exercise generation module creates faulty versions of the solutions provided the teacher, by injecting them different types of error randomly.

Finally, in the game module, students play the role of owner of a small business. In order to increase the value of the company, students must solve UML class diagrams correctly, obtaining profits in return. With this money, the student may hire employees and buy special bonuses which will increase the value of the next solved diagrams, in order to obtain further benefits and lead the ranking of most valuable companies.

Keywords

Gamification, Object Orientation, UML Class Diagrams, Software Design

Índice general

Índice de figuras

Índice de tablas

Glosario

Agregación	Relación que indica que una clase es parte de otra clase.
Asociación	Relación que define una conexión entre clases.
Atributos	Características de una clase.
Clase	Unidad que contiene toda la información de un tipo de objeto.
Clase abstracta	Clase que no puede ser instanciada.
Composición	Relación que indica que una clase forma parte de otra clase y además la clase contenida será destruida si se destruye el contenedor.
CSS	Cascading Style Sheets. Se trata de un sencillo lenguaje con el que podemos especificar varias propiedades de estilo de los elementos de una página web.
Dependencia	Indica que la instanciación de una clase es dependiente de la de otra.
Diagrama de clases	Diagrama que muestra la estructura de un sistema orientado a objetos a través de una representación de las clases con sus métodos y atributos y las relaciones que las unen.
Gamificación	Técnica que consiste en aplicar mecánicas de juego a actividades que no son lúdicas, con objeto de hacerlas más atractivas
Herencia	Relación que indica que una subclase hereda atributos y métodos de una super clase.
HTML	HyperText Markup Language. Lenguaje de marcado para la que define una estructura para la elaboración de páginas web.
Interfaz	Colección de métodos abstractos, implementables por clases.
JS	JavaScript. Lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
JSON	JavaScript Object Notation. Un formato bastante ligero empleado para el intercambio de datos basado en la notación para objetos empleada en JavaScript.
MD5	Algoritmo de cifrado de 128 bits ampliamente usado.
Mecánicas de juego	Conjunto de reglas de juego con las que el usuario, al llevar a cabo una acción, produce un cambio en el estado del juego.
Métodos	Funciones de interacción de una clase.
Multiplicidad	Indica el número de elementos involucrados en una relación.
Orientación a objetos	Paradigma de programación que utiliza objetos para el diseño e implementación de soluciones informáticas.
PHP	PHP: Hypertext Processor. Se trata de un lenguaje que se ejecuta en el servidor orientado al desarrollo web de contenido dinámico
SQL	Structured Query Language. Lenguaje estructurado que permite especificar varios tipos de operaciones en bases de datos relacionales.
Visibilidad	Indica desde dónde se puede acceder a un atributo o método.

1. Introducción

1.1 Antecedentes

El uso de mecánicas de juego para ayudar al aprendizaje no es algo nuevo. Un ejemplo de esto sería el ajedrez, que durante la edad media era usado para enseñar estrategia militar. A pesar de esto, hasta hace unos años no existía el término gamificación ni tenía la importancia que está cobrando actualmente.

Llegados a este punto, definamos qué es la gamificación [1]: Según definió Nick Pellin cuando acuñó el término en 2002, consiste en el uso de mecánicas de juego para motivar a usuarios a llevar a cabo determinadas actividades o comportamientos. Las mecánicas de juego son una serie de reglas con las que el usuario produce un cambio en el estado del juego al llevar a cabo una acción.

En parte, este interés por la gamificación se ha visto impulsado por la creciente industria de los videojuegos, que al comenzar a tener tanta importancia ha creado la necesidad de profesionales especializados. Para poder ofrecer esta preparación, comienza el estudio académico de los videojuegos, lo que permite clasificar e identificar los tipos de juegos, los tipos de jugadores y los objetivos y mecánicas de los videojuegos entre otros.

Gracias a esto y al estudio psicológico de los jugadores y las motivaciones que les llevan a invertir tanto esfuerzo y tiempo en los videojuegos, la gamificación se ha visto impulsada y ha comenzado a ser usada no sólo en medios educativos sino también por compañías de marketing y por comunidades para hacer a sus miembros más activos.

Examinemos ahora algunas de las mecánicas de juego más usadas en la gamificación:

- **Puntuación y recompensas:** El usuario al llevar a cabo determinadas acciones obtiene puntos o recompensas.

- **Desafíos:** Los usuarios compiten en un ranking o directamente entre ellos para obtener sus recompensas.

- **Niveles**: El usuario va desbloqueando actividades según lleva a cabo otras.

- **Misiones:** Se establecen objetivos que el jugador debe ir llevando a cabo.

Una prueba de que la gamificación se va abriendo paso es ver cómo empresas, plataformas y comunidades van implantando estas técnicas. Algunos ejemplos relevantes de su eso en el ámbito educativo son las siguientes:

- **Intermatia** [13]: Es una plataforma web desarrollada para ayudar a los alumnos de instituto a mejorar su nivel en matemáticas. El alumno tiene que resolver una serie de problemas que al resolver correctamente le van dando acceso a niveles superiores.

- **Coursera** [9]: Se trata de una plataforma web didáctica que pone a disposición de los usuarios formación gratuita. Estos cursos consisten en videos que una vez visualizados

al completo dan lugar a un test. Cuando el usuario completa los cursos obtiene un aumento de nivel y si ha logrado lo necesario insignias.

- **The World Peace Game** [25]: Un elaborado simulador político donde un grupo de alumnos tiene que explorar un mundo que comparten. El objetivo consiste en llegar a una situación de armonía entre naciones con mínima intervención militar, lo que les hace aprender la importancia y la complejidad de las relaciones entre países.

- **ClassDojo** [6]**:** Un juego que fomenta el buen comportamiento en clase y el compañerismo. En él, los alumnos tienen un avatar que obtiene puntos y logros por ayudar a otros o por llevar conductas adecuadas.

Viendo que la gamificación se ha aplicado con éxito en el ámbito educativo, surgió la idea de este proyecto, con el cual se pretende facilitar a los alumnos (cuyo temario esté relacionado con la programación orientada a objetos) a desenvolverse mejor con la resolución de diagramas de clases.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una plataforma completa en la que, mediante técnicas de gamificación, los alumnos puedan mejorar de manera autónoma y entretenida sus conocimientos de orientación a objetos y diagramas de clases UML.

Como se pretende que esta plataforma sea usada por el mayor número posible de alumnos y esté siempre disponible, se accederá a la misma a través de una página web, de tal modo que no haya que preocuparse por el funcionamiento en diferentes máquinas y tan sólo se necesite un navegador para acceder a los ejercicios.

La parte principal del proyecto será un juego donde el alumno encarne el papel del dueño de una empresa que tiene que resolver diagramas de clases para obtener beneficios y hacer crecer su empresa. Estos beneficios se obtendrán al resolver problemas generados aleatoriamente a partir de diagramas correctos, previamente definidos por un profesor. Cuando un alumno da por finalizada la resolución de un problema obtendrá una corrección que le ayude a darse cuenta de los errores cometidos. Después de esto y al obtener los beneficios, el alumno entra en un ranking donde se clasifican las empresas que más ganancias han tenido, de esta forma se explota la competitividad de los alumnos.

Además del sistema de juego, también se ofrecerá un sistema de gestión para profesores que permita cómodamente gestionar los usuarios de los alumnos y crear usuarios para otros profesores.

Por último, también se construirá un editor web de diagramas de clases que se usará tanto en el sistema de gestión para que los profesores puedan crear los diagramas a partir de los cuales se generarán automáticamente los problemas del juego, como en el juego para que los alumnos puedan resolver los problemas. Este editor debe permitir crear elementos del diagrama de la forma más flexible posible.

1.3 Estructura del documento

El resto del documento está estructurado de la siguiente forma:

En la Sección 2, se comenzará con un estudio del estado del arte, es decir, analizando algunos de los ejemplos de gamificación con uso educativo más conocidos que existen actualmente, así como de las tecnologías disponibles para llevar a cabo este proyecto.

En la Sección 3: Análisis, se proporciona a una explicación del análisis llevado a cabo y de los requisitos del sistema.

En la Sección 4: Diseño y Desarrollo, Se explicará por secciones las decisiones tomadas para cada una de las partes del proyecto: El juego, el editor de diagramas, el sistema de gestión, la base de datos, el guardado de diagramas y los elementos de juego. En esta misma sección, se explicarán también cuestiones que se consideren relevantes sobre el desarrollo de cada una de las partes.

Posteriormente, en la Sección 5: Pruebas, se expondrán las pruebas realizadas para comprobar la correcta funcionalidad de cada módulo e intentar localizar errores que pueda haber en el sistema.

Finalmente, en la Sección 6, se expondrán las conclusiones, donde se hará referencia a todo lo aprendido con la realización del proyecto y una valoración sobre el uso de la gamificación como herramienta aplicada al aprendizaje del diseño orientado a objetos. Además se incluirá una lista de posibles extensiones a la implementación realizada.

En los anexos se puede encontrar un manual de usuario y otro de administrador aclarando dudas sobre el funcionamiento de diferentes partes de la plataforma.

2. Estado del arte

En esta sección se va a hacer un estudio del estado actual de la gamificación junto a un análisis de algunas plataformas de éxito, lo que nos ayudará a entender mejor qué es la gamificación y cuán extendido está su uso.

Tras este estudio general se expondrá el motivo por el cual se seleccionaron determinadas librerías y herramientas para el proyecto.

2.1 Gamificación en la actualidad

Algunos estudios calculan que sólo en Estados Unidos alrededor de treinta y cuatro millones de personas invierten una media de veintidós horas semanales en videojuegos [23].

Al examinar estos datos es fácil preguntarse qué es lo que hace a las personas dedicar tanto tiempo a algo que realmente no les reporta beneficio alguno e incluso a veces les hace invertir dinero. Y en base a esta pregunta se comenzó a analizar las motivaciones de los jugadores [2]:

- **Recompensas:** Los jugadores obtienen normalmente una recompensa por llevar a cabo una acción: puntos, objetos específicos del juego, puntos de experiencia, etc.

- **Reconocimiento:** Se observó que muchos jugadores formaban comunidades o grupos dentro de los propios juegos y entre ellos mismos se reforzaban continuar en el juego, por sentimiento de compromiso con un equipo o por la amistad generada entre los miembros en el juego.

- **Diferenciarse:** Una gran cantidad de personas siente la necesidad de diferenciarse del resto, de que se reconozcan sus méritos y de ser valorados. Los jugadores buscan esto mismo pero principalmente en comunidades creadas dentro de los propios juegos.

- **Competición:** La motivación de esforzarse, mejorar uno mismo y superar a otros. Esto se puede llevar a cabo tanto con una clasificación como enfrentando a los jugadores entre sí.

- **Logros:** También se observó que algunos jugadores buscaban completar el juego, terminar las misiones o lograr una serie de objetivos.

Entonces se pensó en extrapolar estas motivaciones a entornos no lúdicos y de esta forma hacer que tareas o acciones, que normalmente puedan resultar tediosas o monótonas, sean llevadas a cabo con mucha más facilidad, rigor y compromiso.

La idea de gamificación siempre había estado presente, pero hasta hace bien poco no había recibido un nombre [1]. Como se ha mencionado anteriormente, la gamificación ha visto incrementado su uso, en gran parte, al unirse la industria del videojuego y el ámbito académico, es entonces cuando se empieza a formalizar la idea.

Sin duda el primer sector en darle un uso práctico fue el académico. Pero, ¿qué sectores se benefician más actualmente de esta técnica?

Poco a poco, compañías de publicidad y marketing vieron que podrían beneficiarse al aplicar esta técnica. Y tras ellos, muchos otros empezaron a aplicarlo. Hoy en día, algunos de los sectores donde la gamificación se ha hecho muy popular son:

- **Educación:** Algunos alumnos pueden encontrar el proceso de aprendizaje tedioso. Gracias a la gamificación se puede hacer que un alumno aprenda incluso sin que se dé cuenta de que lo está haciendo. Quizá este sector es en el que más abundan los ejemplos y más variados son. Hay por ejemplo aplicaciones que usan la gamificación para enseñar idiomas, como hace Duolingo [11], una aplicación móvil con más de diez millones de descargas, en la que el usuario sube de nivel al completar ejercicios y funciona con un sistema de vidas que se pierden con las respuestas incorrectas. En la introducción se mencionaron otros ejemplos en este ámbito, como Intermatia [13], Coursera [9], World Peace Game [25] y ClassDojo [6].

- **Empresa:** El método se ha usado para mejorar el trabajo en equipo, para formar a los empleados e incluso para conseguir que el empleado realice un trabajo extra. Por ejemplo, Correos propuso a los empleados revisar las más de 160.000 páginas de su sitio web y proponer mejoras [9]. A cambio los empleados recibirían puntos. Con estos puntos rivalizarían con otros empleados y los que más puntos tuvieran obtendrían un premio. Se calcula que gracias a esto la empresa se ahorró un 70% de lo que costaría que una consultora hiciera este trabajo. Se calcula que en 2015 más del 40% de las empresas utilizarán la gamificación en su beneficio.

- **Marketing:** Donde ha sido usado para atraer audiencias, introduciendo mecánicas de juego en el diseño de sus productos o servicios para minimizar los tiempos de adopción y aumentar la participación de los clientes, haciendo el producto más atractivo. BBVA Game [4] está considerado uno de los mayores éxitos de la gamificación en España. Gracias a este sencillo sistema de puntos y recompensas, el BBVA consiguió en solo 10 meses multiplicar por 15 las visualizaciones de su web, aumentar un 1,6 % el tiempo de estancia de los usuarios y quintuplicar sus seguidores en redes sociales.

- **Salud:** Pain Squad Mobile App es una aplicación móvil creada por un hospital en Toronto, diseñada para que los niños informen de sus dolores a través de un juego de rol y aporten datos que los investigadores puedan usar en su seguimiento [18]**.** Pero podemos encontrar ejemplos mucho más sencillos: Zombie, Run! [26] es una aplicación móvil que anima a sus usuarios a correr con objetivos. Introduce al usuario en un contexto en el que tiene que huir, conseguir objetos o completar misiones.

- **Comunidades:** Algunas redes sociales y comunidades han comenzado a aplicar técnicas de gamificación para aumentar la actividad y el compromiso de sus miembros. Ejemplo de este caso sería Steam [22], la plataforma de distribución digital, que incluye logros, niveles e insignias. Este sistema gamificado no sólo sirve para aumentar las ventas, sino que también premia a los usuarios por comentar, hacer críticas y participar en general.

- **Bien social:** También se ha llegado a usar para motivar a la gente a ayudar a diferentes organizaciones sin ánimo de lucro o simplemente a prestarse a ayudar. Un ejemplo de esto sería Stack Overflow [21], un sitio web donde los desarrolladores web pueden compartir sus conocimientos, y obtienen una serie de insignias en base a lo que puntúen de estos otros usuarios.

En el siguiente gráfico se puede observar lo que se invierte en gamificación en cada uno de estos sectores.

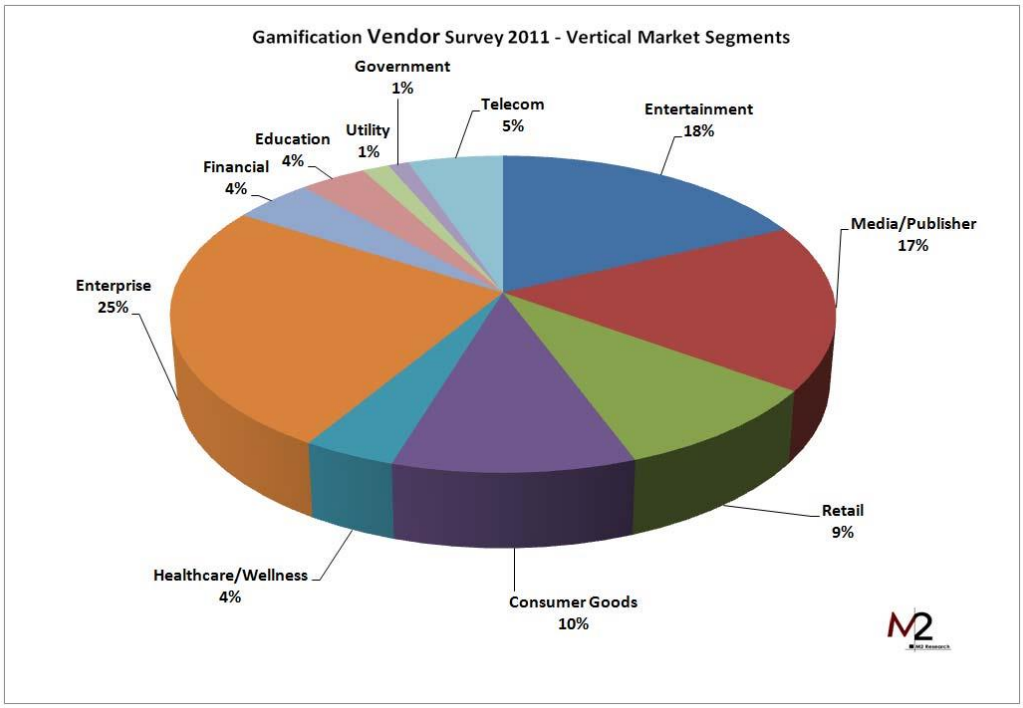


Ilustración 1: Inversión en gamificación de los diferentes sectores [15]

La inversión total realizada en gamificación ha ido creciendo exponencialmente desde el 2011. Se prevé que con el éxito que está teniendo, en 2016 serán invertidos entre todos los sectores aproximadamente 2.8 mil millones de dólares en Estados Unidos en gamificación [14].

2.2 Análisis de las tecnologías usadas

En esta sección se va a explicar las librerías y tecnologías que han sido investigadas y el motivo por el cual fueron elegidas para este proyecto: Raphael para la representación de elementos gráficos; Bootstrap, Tooltipster y Animate.css para ofrecer un aspecto visual atractivo; DataTables para los listados de elementos del juego, usuarios y administradores del sistema de gestión. Por toda la funcionalidad que aporta se utilizó JQuery a la que se añadió un plugin para generar cadenas md5 y así poder tener cifradas las contraseñas de los usuarios y administradores.

Pero antes de analizar en mayor profundidad las librerías y los frameworks, se hará una breve explicación de las tecnologías con las que el proyecto se ha implementado:

Al ser una aplicación web toda la parte de usuario será construida con **HTML**, **JS** y **CSS**. De estos tres lenguajes, el más importante para el proyecto es JS, con el que se implementará toda la parte de procesamiento del juego y del editor de diagramas.

El procesamiento de lado del servidor se llevará a cabo con **PHP**. Este lenguaje se verá mezclado con HTML para generar dinámicamente páginas, como por ejemplo rellenar datos de

las tablas del sistema de gestión con información obtenida con PHP de la base de datos. También implementará servicios que pueden ser invocados con Ajax, como funciones de guardado o comprobaciones de datos.

Para el almacenamiento en ficheros se ha optado por **JSON**. JSON es un simple lenguaje basado en la representación de objetos de JS. Se eligió por ser un poco más legible que otros lenguajes, por la sencillez con la que se puede transformar estructuras de JS a JSON y por las funciones de manipulación y lectura de objetos JSON que tiene PHP.

Una vez explicados los lenguajes que se han utilizado en el proyecto pasemos a las librerías y frameworks utilizados.

Ya que toda la acción del proyecto gira en torno a la edición de diagramas de clases, es quizá de las decisiones más importantes elegir una buena librería JavaScript para gráficos que nos permita obtener un buen resultado.

Se pueden encontrar gran variedad de estas librerías, pero finalmente se optó por **Raphael**. Esta librería no sólo nos permite representar el diagrama de forma sencilla, sino que también incluye animaciones y funciona en los navegadores antiguos, ya que usa VML junto con SVG. Además está orientada a objetos, de forma que será sencillo guardar una relación entre el objeto que es la clase y el objeto que es el dibujo que la representa. El mayor inconveniente de esta librería es que no se tiene el movimiento, rotación y escalado de los objetos automatizados, pero tras un poco de investigación se vio que implementar esto sería sencillo. Otro motivo decisivo fue la gran cantidad de documentación existente y la gran comunidad que hay en torno a esta librería.

En la parte de juego la librería Raphael también será usada para representar en pequeños canvas los modificadores que vaya adquiriendo el usuario.

JQuery [16] fue incluido también en este proyecto, aparte de porque fuera necesario para algunas de las librerías que se han usado, por la simplicidad, sencillez y fácil redacción que añaden a JavaScript. Las peticiones AJAX también jugarán un papel importante en todo el proyecto. Se utilizarán para realizar todo tipo de guardados de datos y comprobaciones desde el cliente. Hacer peticiones que actualizan el contenido del cliente sin necesidad de recargar la página dará mayor sensación de fluidez a la página. La versión de JQuery usada es la 1.11.1, ya que cubre los mínimos que necesitan todas los plugins añadidos al proyecto.

Es muy importante que la aplicación ofrezca un aspecto atractivo para los usuarios. Para ello se eligieron las siguientes librerías: Bootstrap, Animate.css y Tooltipster.

Bootstrap [5] es un framework muy popular que nos permitirá ahorrar mucho tiempo en la implementación del aspecto visual de los elementos, ya que incluye una serie de estilos y funcionalidades JavaScript que nos harán mucho más sencillo todo. Bootstrap se utilizará para toda la aplicación, tanto en el sistema de gestión como en el juego, para ofrecer un aspecto más profesional. Por ejemplo Bootstrap añade un sistema de cuadrícula con el que podemos definir las columnas y las filas, lo que no nos evita preocuparnos de alturas y anchuras, ya que se definen los elementos en paneles ajustados a esta cuadrícula. Bootstrap también hace que una página sea mucho más fácil de adaptar a dispositivos móviles.

Animate.css [3] incluye un conjunto de animaciones ya implementadas en CSS que nos ahorrará tiempo. Por ejemplo se usarán animaciones como Shake (que hace a los elementos

dar una pequeña sacudida), Flash (que hace parpadear un elemento) y efectos de entrada y salida de elementos. Shake podrá ser usado cuando se deniegue la acción del usuario, por ejemplo al fallar el acceso o al no poder adquirir un elemento del juego por falta de recursos. Flash será utilizado para llamar la atención de un usuario sobre un elemento, como por ejemplo, el botón de realizar problemas. Los efectos de entrada y de salida (de los cuales hay gran variedad) serán utilizados para dar sensación de fluidez en la aplicación en general.

Tooltipster [24] es un plugin para JQuery que nos permite personalizar los tooltips (o información emergente) e incluso hacerlos interactivos, cosa que será muy útil a la hora de implementar el juego. En el juego contendrán información sobre los elementos modificadores, que aportarán al usuario un extra de puntos conseguidos por resolver diagramas. Además de la información sobre el elemento, tendrán un botón que permita adquirir el elemento.

También se ha tenido en cuenta que el sistema tendría varias tablas para contener información: usuarios, administradores, estado del juego y elementos del juego. Así que resultaría muy útil tener unas tablas lo más flexibles posibles, con posibilidad de realizar búsquedas y de ordenar por cualquier columna. Para facilitar la implementación de esta parte se pensó en usar **DataTables** [10], que es otro plugin para JQuery que permite hacer todo esto de forma muy sencilla.

Por último se añadió un plugin, **JQuery-MD5** [16], para cifrar las contraseñas de los usuarios y administradores a md5 y hacer el sistema un poco más seguro. De esta forma las contraseñas se cifran en el cliente y se envía la cadena cifrada para el acceso y para el registro, quedando almacenada en la base de datos no una contraseña visible, sino una cadena MD5.

3. Análisis

En este aparatado se realizará un análisis de requisitos de las tres partes principales de Diagame: el editor de diagramas, el sistema de gestión y el juego.

3.1 Análisis del editor de diagramas

Comenzaremos el análisis con la parte en torno a la cual gira todo el proyecto: los diagramas de clases. Es importante que el editor de diagramas ofrezca una funcionalidad lo más completa posible, para permitir a los profesores crear diagramas de la forma más flexible posible, como si lo hicieran sobre papel. Los requisitos para el editor de diagramas son los siguientes:

- **Clases:** Crear clases con sus atributos y métodos. Por un lado en los atributos se debe poder definir la visibilidad, el nombre, el valor por defecto y el tipo. Por otro se debe poder definir la visibilidad, el nombre, los argumentos y el tipo de valor devuelto de los métodos.

- **Relaciones:** Crear relaciones de los tipos: herencia, asociación, agregación, composición y dependencia. Además para asociaciones, agregaciones y composiciones se debe poder añadir la multiplicidad, la navegación, y el rol de cada clase si es que los tienen. En el caso de las dependencias también se debe poder establecer un nombre para la dependencia.

- **Unión automática:** La ruta que sigan las relaciones debe ser generada automáticamente en función de la posición de las clases, incluso si éstas son desplazadas.

- **Selección de estrategia de unión:** El punto de unión de una clase con la relación podrá ser definido manualmente por el usuario cada vez que cree una relación o bien seleccionar una estrategia por defecto: mitad, aleatoria o personalizada. Si se selecciona la estrategia de mitad, la relación partirá del punto medio del lado de la clase desde donde parta la relación. La estrategia aleatoria escogería un punto al azar de la clase, lo que resulta útil para impedir que se solapen varias relaciones que partan de una clase. Por último en la estrategia personalizada el usuario tendrá que definir para cada relación los puntos que la unen con cada clase.

- **Edición y borrado:** se debe poder modificar o borrar cualquier elemento del diagrama.

- **Tamaño variable:** Los diagramas podrán tener tamaño variable definido por el usuario.

- **Guardado y cargado**: Los diagramas se almacenarán en el servidor y podrán ser cargados para ser editados de nuevo. También existirá una opción que permita cargar un diagrama sobre uno ya existente, permitiendo de esta forma combinar diagramas sin necesidad de copiar a mano uno sobre otro.

- **Modo tutorial:** El diagrama tendrá una opción de modo tutorial. Al estar activa esta opción aparecerán mensajes sobre el diagrama ayudando al usuario a hacer uso del mismo.

- **Enunciados**: Cuando el profesor guarda el diagrama debe poder añadir un enunciado o descripción al diagrama. Esta descripción le será mostrada a los alumnos para ayudarle a resolver los problemas generados a partir de este diagrama.

3.2 Análisis del módulo de juego

En cuanto al juego, es importante que se vean aplicadas claramente mecánicas de juego para llevar a cabo el proceso de gamificación correctamente. El juego se inspirará en el juego Cookie Clicker [7], que consiste en ganar puntos (galletas) haciendo click en un objeto. Con estas galletas se pueden comprar mejoras, que hacen que al hacer click en un objeto ganemos más galletas o que se produzcan galletas de forma constante automáticamente.

En el juego implementado para Diagame, el alumno interpretará el papel de dueño de una pequeña empresa que tendrá que dirigir para convertirla en la mayor de todas. Pero esto no será una tarea sencilla, pues hay otros alumnos que hacen crecer sus propias empresas con el mismo fin. Para aumentar el valor de la empresa, el usuario tendrá que aumentar el número de empleados de ésta y llevar a cabo medidas temporales para incrementar los beneficios.

El dinero necesario para llevar a cabo todas estas inversiones se obtendrá realizando proyectos. Estos contratos consisten en resolver diagramas de clases UML que tienen algún problema.

Las inversiones en personal se amortizarán a la larga, pues cada empleado aportará mayor capacidad a la empresa, lo que se representará en un pequeño incremento por empleado de los beneficios de cada proyecto. Existen diferentes tipos de empleados (por ejemplo: becarios, programadores y analistas) que aportan diferentes modificadores pero que tienen diferentes costes de contratación. En adelante nos referiremos a los empleados como modificadores constantes.

Por otro lado tenemos las medidas temporales, a las que nos referiremos también como modificadores temporales. Estos modificadores son de un solo uso. Es decir, el usuario tiene que activarlos antes de cada contrato si quiere que se aplique el modificador a los beneficios que obtenga. La principal ventaja de estos modificadores temporales es que aportan un modificador a los beneficios mucho mayor que los modificadores constantes, aunque también son bastante más costosos.

Para ambos tipos de modificadores se aplica la misma regla: cuantos más elementos del mismo tipo se hayan adquirido para la empresa más costoso resultará. Por ejemplo, contratar el segundo becario es más costoso que contratar al primero. El tercer becario costará más todavía que el segundo, y así sucesivamente.

Cuando un alumno da por terminada la resolución de un diagrama obtendrá un resumen de los errores cometidos y de los beneficios obtenidos finalmente. Cada error cometido añade una pequeña cantidad al valor total de los errores cometidos. El valor de la cantidad añadida por error varía dependiendo del tipo de error. Por ejemplo añadirá más valor al error total que al diagrama dado como respuesta le falte una clase o una relación, de lo que añadiría la visibilidad incorrecta de un atributo.

Por cada punto acumulado como error total el usuario perderá un 20% del valor que supondría el beneficio inicial del diagrama. De esta forma si el valor del error supera la cantidad de 5, el

usuario perdería un 100% y no obtendría beneficios, con lo que el proyecto se consideraría fallido.

Los beneficios para cada proyecto se calculan en base al número de elementos que tenga. Al igual que con los errores, los diferentes elementos del diagrama aportan diferentes cantidades al valor del diagrama. Cada clase y cada relación añaden más valor al diagrama que cada atributo o método. Con esta valoración se pretende valorar la dificultad que pueda tener un diagrama, ya que al generarse aleatoriamente alteraciones sobre cada elemento del diagrama en los problemas, pueden darse más errores en un diagrama con más elementos.

Según el avance en el juego, el alumno puede ver el valor de su empresa frente a las de sus compañeros en todo momento, así como una lista de las 10 empresas más valiosas. Para incentivar a los alumnos a competir y mejorar su empresa, lo ideal sería que los profesores añadiesen alguna recompensa al mejor o mejores del ranking al terminar el curso. De esta forma se motivará a los alumnos a realizar diagramas de clases UML y mejorar su técnica.

Los requisitos a la hora de implementar esto son los siguientes:

- **Mecánicas de juego:** En el sitio web se deben usar mecánicas de juego para incentivar que los alumnos resuelvan problemas en los que un diagrama de clases alterado tenga que ser corregido por el alumno. Se tendrá un ranking de usuarios (Competitividad) y el dinero virtual invertido en la empresa, que es una puntuación (Recompensas) por los ejercicios resueltos correctamente. También habrá una lista pública de los usuarios con mejor puntuación (Reconocimiento).

- **Problemas autogenerados**: Los problemas serán generados automáticamente por el sistema y de forma aleatoria, de forma que cada vez que acceda, el alumno se enfrentará a un problema diferente. Los problemas se generarán a partir de soluciones correctas proporcionadas por el profesor.

- **Añadir generadores de problemas:** Se debe poder añadir al sistema generadores personalizados que produzcan los tipos de problemas que desee el profesor. El sistema tendrá los siguientes generadores básicos implantados por defecto: Alteración de la visibilidad de los atributos y métodos, cambios de sentido en las relaciones e intercambios en el tipo de algunas relaciones (asociaciones, composiciones y agregaciones) y borrado de relaciones.

- **Editor de diagramas**: El editor de diagramas usado por los alumnos será el mismo que el de los profesores, solo que con algunas funcionalidades anuladas (Como por ejemplo editar la descripción de un diagrama).

- **Retroalimentación:** Una vez el alumno ha terminado de resolver un problema, el sistema debe mostrar los errores que haya tenido, ayudando a evitar que los cometa en diagramas futuros. Al usuario se le premiará, cuando resuelva correctamente el problema, con dinero virtual para invertir en su empresa y subir de posición en el ranking.

- **Acceso al sistema:** Se accederá con un usuario y contraseña por alumno, de forma que el profesor pueda seguir qué alumnos son los que más practican y los que menos. Estos datos se mostrarán al profesor en una tabla que se puede ordenar por columnas y que permita búsquedas para hacer filtrados de los datos.

- **Aspecto visual (requisito no funcional):** Que la aplicación resulte atractiva y tenga un aspecto vistoso es fundamental para hacer que los usuarios se sientan atraídos a hacer uso de la aplicación.

3.3 Análisis del módulo de gestión

Con el sistema de gestión se tiene que proveer a los profesores una herramienta con la que personalizar el juego y gestionar los usuarios tanto del juego como del propio sistema de gestión. Más concretamente este sistema debe permitir lo siguiente:

- **Gestionar usuarios para los alumnos:** El sistema permitirá la edición, creación y borrado de usuarios de alumnos, que pueden acceder al juego pero no al sistema de gestión.

- **Gestionar diagramas de clases:** Los profesores tendrán acceso al editor de diagramas de clases para generar diagramas a partir de los cuales se crearán los problemas para los alumnos

- **Gestionar usuarios para otros profesores:** También se debe poder permitir la edición, creación y borrado de usuarios profesores, los cuales pueden acceder al sistema de gestión, pero no al juego.

- **Gestionar los elementos que aparecerán en el juego:** Se permitirá crear y eliminar los elementos que formen parte del juego (es decir, los diferentes tipos de elementos con los que podrá interactuar el usuario en el juego, como por ejemplo un tipo de modificador que pueda adquirir), personalizándolo de esta forma a gusto del profesor. Estos elementos también se almacenarán en formato JSON

- **Fácil de compartir:** Los diagramas y elementos deben ser fácilmente exportables para que si un profesor de un centro quisiera compartirlos, se pueda llevar fácilmente de un sistema a otro.

4. Diseño y desarrollo

En este apartado se hará una breve explicación general de la arquitectura del sistema. Después de esto, se procederá a explicar punto por punto las partes relevantes en el diseño y desarrollo de cada parte del sistema.

4.1. Arquitectura del sistema

Como se ha explicado anteriormente, Diagame se divide en tres subsistemas. El sistema de gestión tiene como fin permitir a los profesores gestionar tanto los usuarios como el juego. Por otro lado el juego, es la parte donde acceden los alumnos y donde se pone en práctica las técnicas de gamificación.

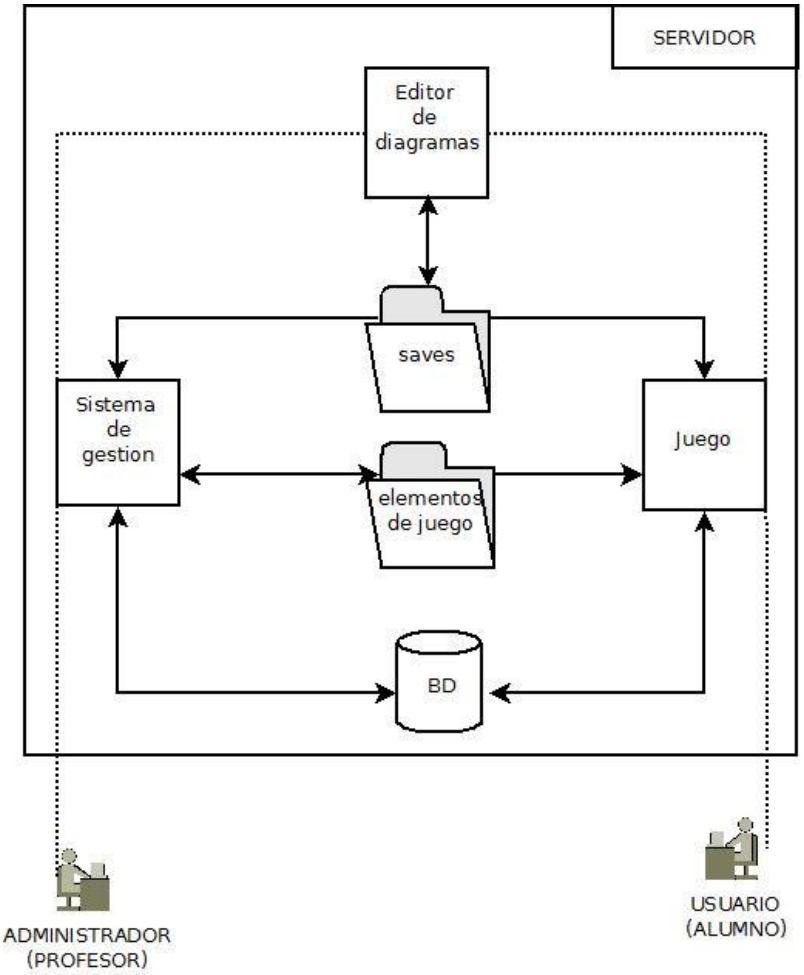


Ilustración 2: Arquitectura del sistema

Los sistemas se han diseñado para que sean independientes el uno de otro, de forma que los únicos ficheros de código que comparten son los de las librerías y frameworks externos (JQuery, Bootstrap, etc). Como se puede ver en el esquema, los dos sistemas comparten tres elementos: una base de datos MySQL y dos carpetas donde se almacenarán ficheros JSON. Las flechas indican el tipo de acceso de un sistema a uno de estos elementos compartidos: las

flechas dobles indican lectura/escritura y las que van solo de un elemento al sistema indican lectura únicamente.

En la base de datos se almacenan la información de los usuarios y de los administradores, además del estado de cada una de las partidas existentes en el juego. La base de datos es leída y escrita desde las dos partes del sistema.

La carpeta “saves” contiene los diagramas de clases UML que se crean, editan y eliminan desde el sistema de gestión y que se leen desde el juego para generar los problemas que resolverán los alumnos.

La carpeta “elementos del juego” está dividida en dos subcarpetas, una para modificadores y otra para bonus. En la carpeta “modificadores” se almacenarán los elementos que modifiquen los beneficios del jugador al resolver un diagrama de forma permanente. Por otro lado, en la carpeta “bonus” se almacenarán aquellos elementos que modifiquen los beneficios del jugador pero que tienen un único uso. Ambos tipos de elementos son creados, editados y borrados desde el sistema de gestión y ambos tipos son leídos para desde el juego para su integración como elementos de juego.

4.1 Módulo de Juego

4.1.1 Diseño del módulo de juego

A continuación se explicarán algunos aspectos interesantes desde el punto de vista de la programación en el juego. Comencemos viendo cómo es la navegación entre pantallas:

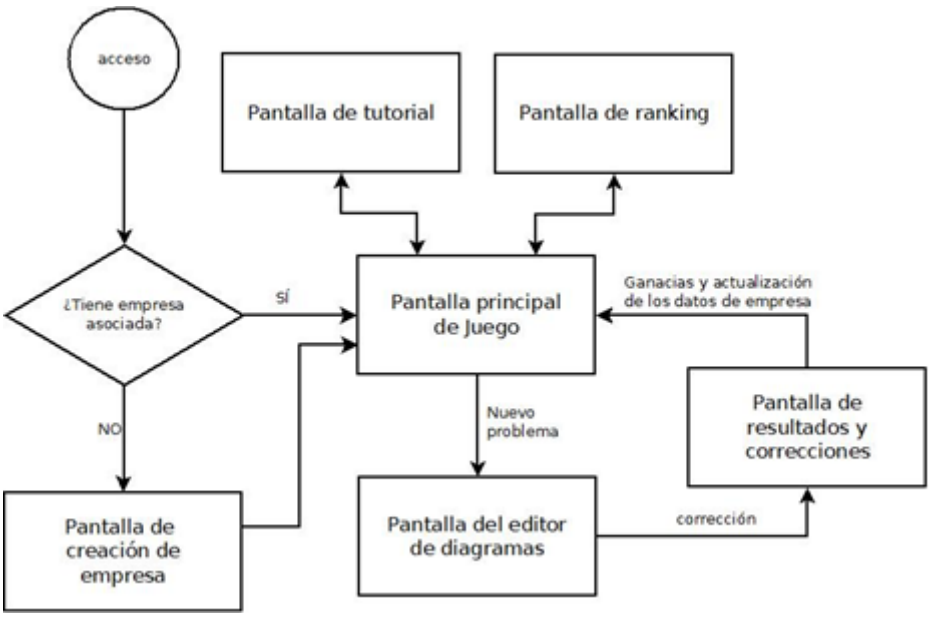


Ilustración 3: Navegación entre pantallas de juego

En este esquema se explica de forma breve el ciclo completo que compone el juego y las pantallas a las que tiene acceso el alumno. Como se puede ver, cuando el alumno entra por primera vez tiene que poner un nombre a su empresa. A partir de este punto, el usuario queda relacionado con la empresa en la base de datos. Veamos en detalle cada una de las pantallas que se pueden encontrar a partir de este punto:

4.1.1.1 Pantalla principal de juego

Es, junto con el editor de diagramas, una de las pantallas más importantes. En ella el usuario gestionará la empresa, es decir, contratará empleados, adquirirá beneficios temporales y empezará nuevos contratos. También tendrá disponible toda la información de su empresa (ganancias, valor de la empresa, contratos realizados, etc), así como acceso al ranking de las mejores empresas del juego.

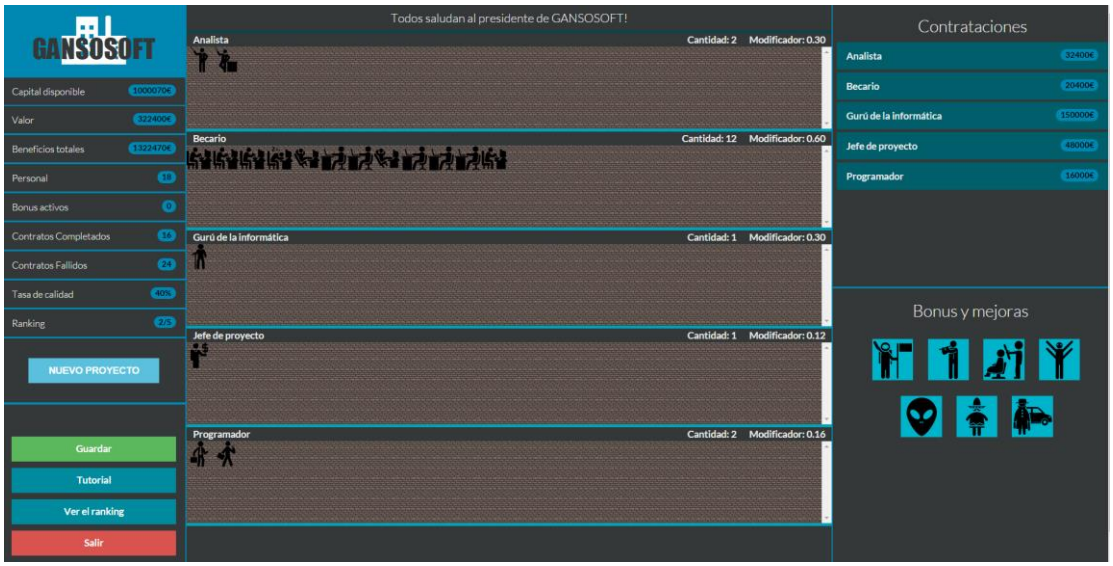


Ilustración 4: Pantalla principal del juego

Como se ve en la imagen, la pantalla principal se divide en tres partes. Empezando por la izquierda encontramos el panel de información de la empresa y los botones de acción, en el centro se encuentran las representaciones de los empleados contratados y por último, a la derecha se encuentran los botones que se usan para contratar empleados y adquirir modificadores temporales. Esta pantalla se adapta a dispositivos móviles (esto se puede consultar en el Anexo 2: Vista del juego en dispositivos móviles).

En el panel de información encontramos una imagen que hace de cabecera con el nombre de la empresa. Debajo, el usuario puede consultar toda la información necesaria de la empresa: el capital disponible para gastar, el valor (dinero invertido) en la empresa, los beneficios totales (dinero disponible más dinero gastado), la cantidad de personal trabajando en la empresa, el número de bonus (modificadores temporales) que se aplicarán al siguiente ejercicio, los contratos terminados con éxito y con fracaso, la tasa de éxito al resolver diagramas del usuario y por último la posición de la empresa entre todas las del juego. Estos elementos se actualizan en tiempo real, salvo el ranking, que se actualiza cada vez que se recarga la página para no tener que estar actualizando constantemente y congestionar el tráfico en el servidor.

Debajo de esta información se encuentran los botones de acción. Los botones de acción permiten al usuario iniciar un proyecto nuevo (es decir intentar resolver un nuevo problema),

guardar los cambios del juego (este guardado se realiza también automáticamente al iniciar un nuevo proyecto), acceder al tutorial, ver el top 10 de empresas y cerrar sesión.

El centro es una zona puramente visual, muestra al usuario una representación del número de cada tipo de empleado que tiene contratado. También le permite ver el modificador total que aportará a los beneficios cada grupo de cada tipo de empleado. Estas secciones se actualizan en tiempo real.

A la derecha encontramos primero la zona de contrataciones, donde el usuario puede añadir empleados (modificadores constantes) a su empresa. Cuando el usuario hace click en uno de los elementos de esta zona surge un tooltip generado con Tooltipster desde javascript. Este tooltip contiene la información del tipo de trabajador seleccionado: Nombre, descripción, precio, el modificador que aporta a los beneficios por unidad, el incremento de precio por cada contratado que haya y un botón para añadir uno de estos modificadores.

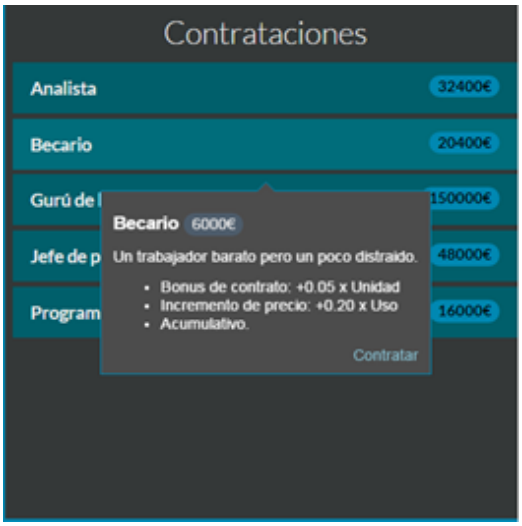


Ilustración 5: Ejemplo de Tooltip de contratación

Si pulsamos el botón comprar se actualizarán inmediatamente el canvas, la información de la empresa y el precio de este elemento. Si se intentase comprar el modificador y no hubiera dinero suficiente, una animación del botón de contratar nos indicaría que no es posible.

Debajo del panel de contrataciones, encontramos el panel de bonus y mejoras. Este funciona de forma muy similar al panel de contrataciones, como se puede ver en la imagen, con la diferencia de que una vez adquirido el modificador temporal cambia de color el botón que lo representa y no se podrá volver a adquirir hasta que se haya gastado (es decir, hasta que se realice el próximo diagrama y se aplique el modificador temporal a los beneficios obtenidos).

Ilustración 6: Ejemplo de Tooltip de bonus y mejoras

4.1.1.2 Pantalla del editor de diagramas

Esta pantalla es prácticamente igual que la explicada en el punto 4.2: Editor de diagramas (donde se explica, con más detalles sobre la implementación), solo que se alteran algunas de las funcionalidades del editor de diagramas, ya que el usuario alumno no necesita algunas de las funcionalidades que el profesor sí. En concreto estas modificaciones son:

- Al cargar la página se añadió un modal que muestra al usuario información del diagrama y del tipo de alteración que ha sufrido el diagrama y del enunciado del diagrama al que se enfrenta. También se añadió un botón para consultar esta información siempre que se quiera durante la resolución del problema.

- Se sustituyó el botón de guardar diagrama por el botón de entregar proyecto, que llama a una función diferente del servidor y que guarda el diagrama como respuesta de un usuario. Este botón lleva al usuario a la página de resultados y correcciones.

Además de todo esto, se sustituyó el CSS del editor de diagramas por uno propio que adaptase el estilo acorde al resto de las páginas del juego.

4.1.1.3 Pantalla de resultados y correcciones

Esta pantalla se muestra al usuario cuando termina de resolver un diagrama de clases. En ella, el usuario encontrará el valor total del proyecto resuelto, es decir, lo que obtendría en caso de que hubiese resuelto el problema a la perfección.

Tras esto encontrará una lista desplegable con los errores que contenga el diagrama entregado. Al final de la página se muestra también el valor de los errores cometidos, si el diagrama se da por fallido o no y los beneficios que resultan finalmente al aplicar los distintos modificadores que tenga la empresa.

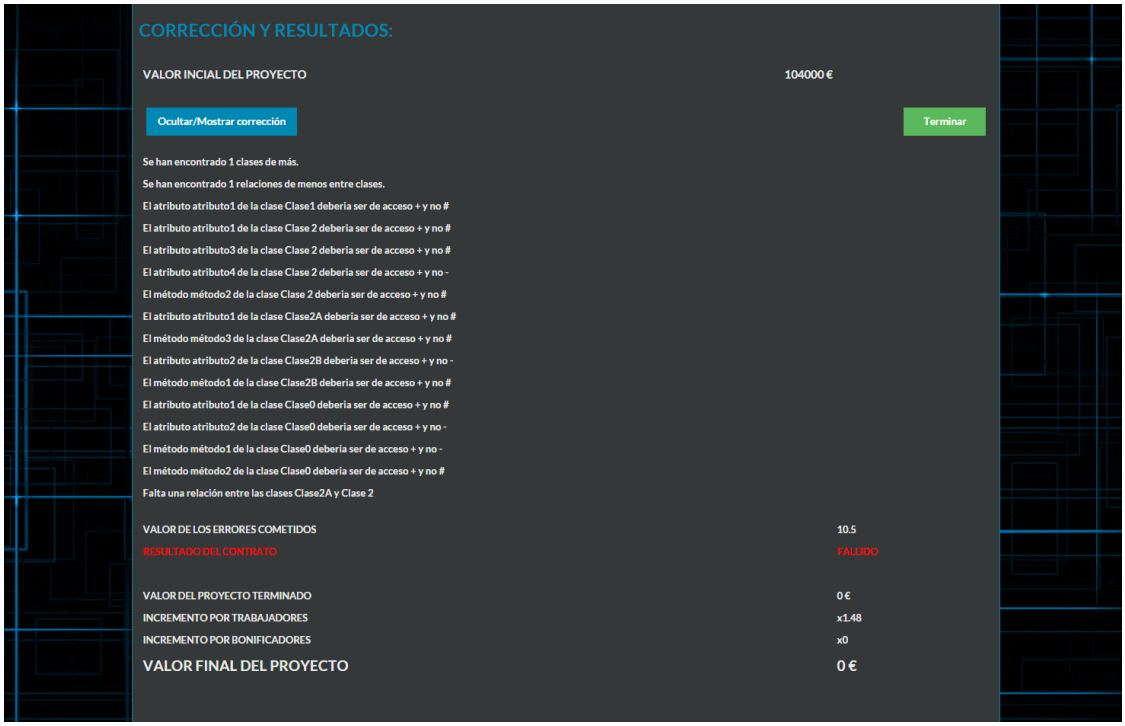


Ilustración 7: Ejemplo de pantalla de correcciones y resultados

Para llevar a cabo la corrección se seleccionan el diagrama que se guardó como solución al abrir la pantalla del editor y el diagrama guardado como respuesta del alumno. Una vez se han obtenido los objetos que representan cada diagrama comienza una comparación. Las diferencias encontradas entre los dos diagramas serán consideradas errores.

Para calcular el valor del error del usuario se clasifican los errores en dos tipos: errores graves y errores leves. Los errores graves añaden 1 al valor del error cometido, mientras que los errores leves añaden 0.5.

La falta o existencia de alguna clase o relación será considerada error grave. También será considerado error grave que los tipos de las relaciones sean diferentes. Por otro lado un error en un atributo o método, o en la multiplicidad de una relación serían errores leves.

El valor obtenido de la suma de todos estos errores, da el valor de los errores cometidos, el cual es multiplicado por 0.2 y por el valor total. El resultado de esta operación es restado al beneficio que aportaría el diagrama originalmente. Si este valor supera 1.0 el beneficio sería cero y la resolución del diagrama se consideraría fallido, obteniendo un beneficio de 0.

Si el usuario ha conseguido obtener beneficios se sumarían los porcentajes correspondientes a los modificadores que posea la empresa al valor total.

En esta pantalla se actualiza el estado de la partida del usuario, guardando en base de datos los valores que han cambiado para su empresa.

4.1.1.4 Pantalla de tutorial y pantalla de ranking

Comencemos con la pantalla del ranking. Esta pantalla, es en realidad un Modal Box que se abre en la propia pantalla de juego, donde se muestra al usuario una tabla de las 10 mejores

empresas ordenadas por orden de valor. Junto al nombre de la empresa se muestra el valor de ésta y el usuario al que pertenece.

En la pantalla de tutorial se accede a una serie de imágenes que muestran al usuario el funcionamiento general del juego. Esta pantalla está pensada para alumnos que entran por primera vez al sistema.

4.1.2 Implementación del módulo de juego

La página se genera con PHP, accediendo a la base de datos para obtener los datos del estado de la partida (información de la empresa y del usuario) y recorriendo las carpetas donde se encuentran los ficheros JSON (consultar el anexo Estructura de los elementos del juego para más información) que forman los elementos de juego.

La lógica y actualización de la interfaz se lleva a cabo con JavaScript, más concretamente en el objeto Juego, que contiene todos los métodos y elementos necesarios para ello.

Toda interacción con el servidor durante esta pantalla se lleva a cabo con AJAX. Mientras se procesan estas peticiones al usuario se le muestra una animación de guardado y se le bloquea la pantalla, hasta que se obtenga la respuesta. En esta pantalla las peticiones que se pueden realizar al servidor son: guardado y nuevo proyecto. La función de guardado invoca un PHP que se encarga de guardar los datos de la empresa en el cliente, actualizando las entradas para esta en la base de datos. Por otro lado, al ser llamado el PHP de nuevo proyecto, se desencadena la creación de un nuevo problema, es decir, se coge al azar uno de los diagramas creados por los profesores y se altera con uno de los generadores disponibles. Se guardan la solución y el problema y se envía al usuario al editor de diagramas, donde se le mostrará el diagrama alterado que tendrá que resolver.

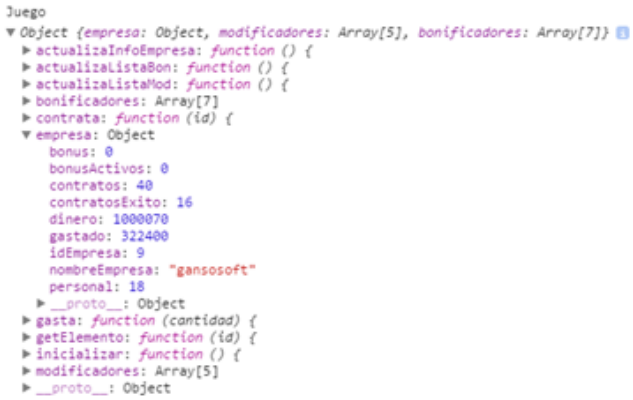


Ilustración 8: Objeto JS juego de la pantalla principal de juego

Al estar los datos de la empresa en JavaScript quedan totalmente a merced del cliente. Esto permite al usuario hacer trampas, alterando manualmente desde la consola del navegador con una sencilla instrucción:

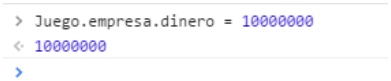


Ilustración 9: Ejemplo de alteración malintencionada desde el cliente

Pero esto se puede controlar sencillamente de lado del servidor, ya que en esta pantalla es imposible variar el valor que se obtiene al sumar el dinero disponible y el valor de la empresa. Así que simplemente, antes de realizar la inserción en la base de datos se podría controlar si cuadran las cantidades recibidas con las previas. Estos controles se deberían aplicar a todos los campos que se actualicen al guardar el estado del juego.

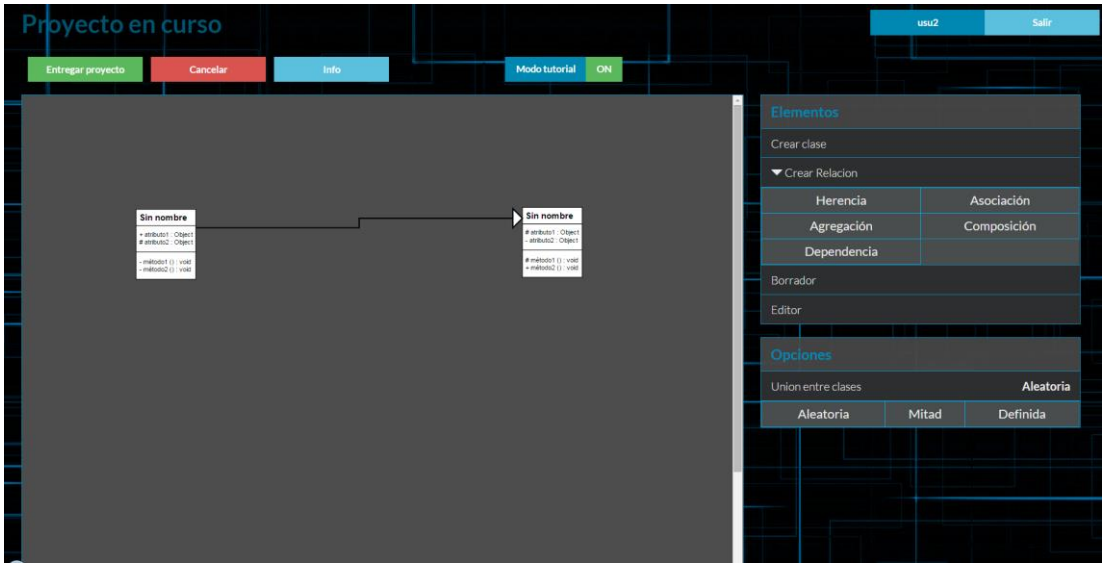


Ilustración 10: Pantalla del editor de diagramas en el juego

Si el usuario abandona esta página en algún momento o si pulsa el botón cancelar o cerrar sesión, el diagrama contará como fallido, lo que no le aportaría ningún beneficio, gastaría los modificadores temporales y bajaría el porcentaje de éxito del usuario.

El diagrama mostrado al usuario en esta página es un diagrama alterado para que sea incorrecto. Para generar este diagrama se selecciona al azar uno de los diagramas creados por los profesores y uno de los generadores de problemas existentes. Se aplica el generador de problemas al diagrama seleccionado y se guardan en una carpeta identificada por la id del usuario el diagrama original y el diagrama alterado. Los generadores son ficheros PHP que contienen código que altera los diagramas. Esto se explica más en detalle en el Anexo 2: Añadiendo Nuevos generadores de problemas.

Por último, cuando un usuario da por terminada la resolución la resolución del problema y pulsa el botón de entregar proyecto, se guarda un tercer diagrama que contiene el diagrama modificado por el alumno en el editor. Tras esto se envía al usuario a la pantalla de resultados y correcciones.

4.2 Módulo del editor de diagramas

4.2.1 Diseño del módulo del editor de diagramas

El editor de diagramas permitirá, por un lado, que los profesores puedan crear y manipular diagramas de clases con los que se generarán problemas para los alumnos, y por otro, a los alumnos resolver los problemas que se generen a partir de esos diagramas.

La lógica del funcionamiento del editor será la misma para alumnos y profesores, con la diferencia de que algunos de los botones serán desactivados para los alumnos, como se explica más en detalle en el punto 4.1.2.2: Pantalla del editor de diagramas. Toda esta lógica está implementada en JavaScript y sólo se interacciona con el servidor con las acciones de guardado y cargado, con las que se invocan servicios PHP y se intercambian objetos JSON.

Al entrar en el editor se le pide al usuario que introduzca el tamaño del canvas donde se va a crear el diagrama. Se ha definido un tamaño máximo, mínimo y por defecto, para que no existan diagramas tan pequeños que no lleguen a cubrir todo el div donde se encuentra situado el canvas ni excesivamente grandes. El tamaño por defecto es usado cuando el usuario no define un tamaño nada más entrar.

Una vez dentro del editor de diagramas, existen una lista de elementos creables y herramientas. Dependiendo de la selección del usuario en esta lista al hacer click en el diagrama se llevará a cabo una acción u otra. Los elementos disponibles en esta lista son:

- **Crear Clase**: Al hacer click en algún punto del diagrama despliega un Modal Box que pide al usuario cierta información. En este modal es donde se definen el nombre de la clase, su tipo (Clase, Abstracta o Interfaz) y los atributos y métodos que posea.

- **Crear relación**: Este elemento abre una sublista en la lista de elementos que muestra todos los tipos de relaciones creables: Herencia, Asociación, Agregación, Composición y Dependencia. Tras seleccionar uno de los elementos de la sublista y hacer click en dos clases del diagrama nos aparecerá un Modal Box con los datos necesarios para crear la relación. Estos datos son diferentes para cada tipo de relación, ya que por ejemplo una herencia no tiene multiplicidad, pero una composición puede tenerla.

- **Borrador**: Tras seleccionar esta herramienta, se borrará todo elemento del diagrama en el que hagamos click.

- **Editor**: Tras seleccionar esta herramienta, se podrá editar cualquier elemento del diagrama en el que hagamos click. Al hacer esto se abriría el Modal Box correspondiente ya rellenado con los datos de la clase o relación, permitiendo al usuario alterarlos, borrarlos o añadir nuevos.

Se puede encontrar imágenes de estos Modal Box en el Anexo 3: Modal Box de elementos del editor de diagramas.

La lista de estos elementos se encuentra situada a la derecha de un canvas junto con una lista de las opciones disponibles. En la parte superior del diagrama se encuentran los botones de nuevo diagrama, de cargado y de guardado. También se encuentra en la parte superior el botón que activa y desactiva el modo tutorial, junto con el botón de cerrar sesión.

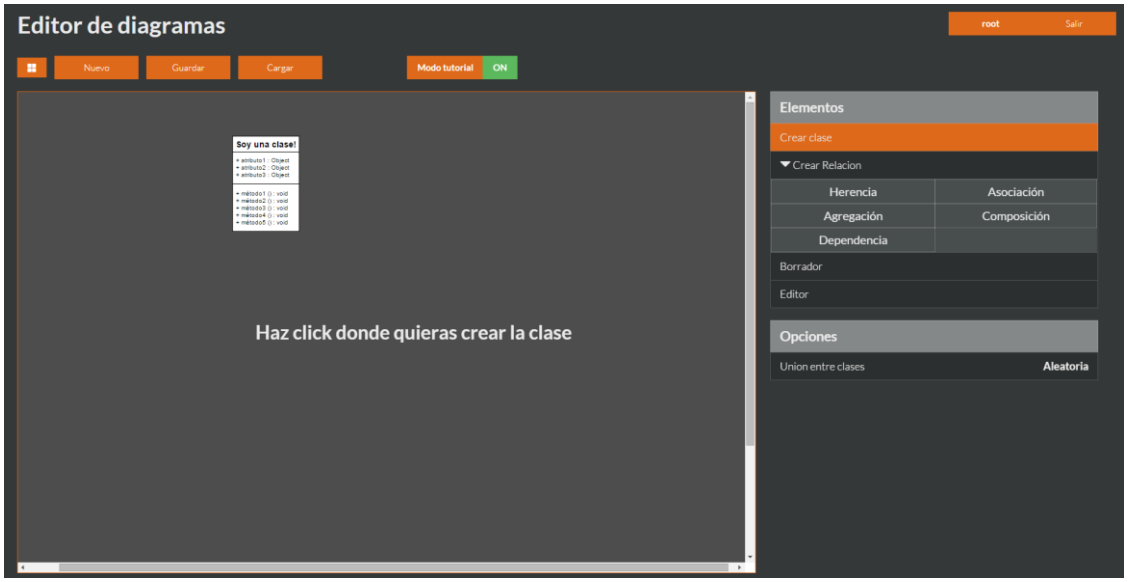


Ilustración 11: Editor de diagramas del sistema de gestión

Si el modo tutorial está activo se irán mostrando mensajes de ayuda para el uso de editor sobre el canvas (como se ve en la imagen). Por ejemplo, si el usuario quiere crear una relación y selecciona en el menú de elementos el tipo de relación herencia, el sistema mostraría un mensaje: “Haga click en la clase padre”. Una vez el usuario ha seleccionado la clase padre, el sistema mostraría: “Haga click en la clase heredera”. Con esto el usuario no tienen necesidad de acudir a ningún manual mientras hace uso de esta parte de la aplicación.

En la lista de opciones, la única disponible es “Unión entre clases”, la cual sirve para establecer el tipo de estrategia de unión que se usará por defecto. Dentro de estos tipos de unión, encontramos: Aleatoria, Media y Definida. Si el usuario escoge la opción Aleatoria, se tomarán puntos al azar de una clase para realizar la unión entre una clase y la relación que la une con otra; En la estrategia Media, el punto de unión seleccionado será exactamente la mitad de la clase; Por último en la estrategia definida, el usuario tendrá que definir manualmente el punto de unión. Esta estrategia puede ser cambiada manualmente al crear una relación si el usuario así lo quiere.

4.2.2 Implementación del módulo del editor de diagramas

En cuanto a la implementación del editor de diagramas, se ha encapsulado todas las funciones y clases en un paquete JS llamado elementosDiagrama. A través de este namespace, se pueden llamar a los constructores de las clases que contienen la información y representación de los elementos del diagrama (clases y relaciones).

En este paquete se guardan también dos arrays que almacenan las instancias de los elementos que existen en el diagrama, uno para las clases y otro para las relaciones. También se almacenan una serie de constantes (como por ejemplo el ancho de las relaciones) y de enumeraciones (como por ejemplo todos los elementos creables).

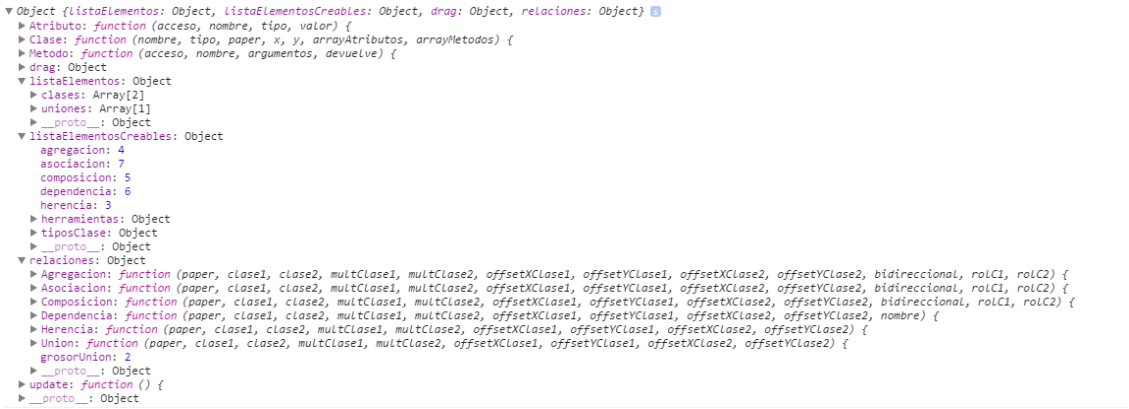


Ilustración 12: Elementos del paquete elementosDiagrama

Cada clase representada en el diagrama se corresponde con un objeto Clase que contienen toda su información y dos arrays: uno de objetos Atributos y otro de Métodos, con lo que cada clase encapsula sus propios atributos y métodos, de forma que si borramos una clase se borran también todos sus objetos Atributo y Método. Cada clase también guarda su representación en el canvas, que consiste en un conjunto o set de elementos (textos, líneas y rectángulos) de la librería Raphael.js.

En el constructor de la clase también se define el comportamiento de arrastrar y soltar que permite mover las clases arrastrándolas con el cursor en el diagrama.

Al igual que con las clases, cada relación del diagrama se corresponde con un objeto JS del paquete. Todos los objetos que representan una relación heredan de la clase Unión, la cual contiene los métodos y atributos comunes a todas las relaciones (como por ejemplo trazar la línea de la relación entre dos clases).

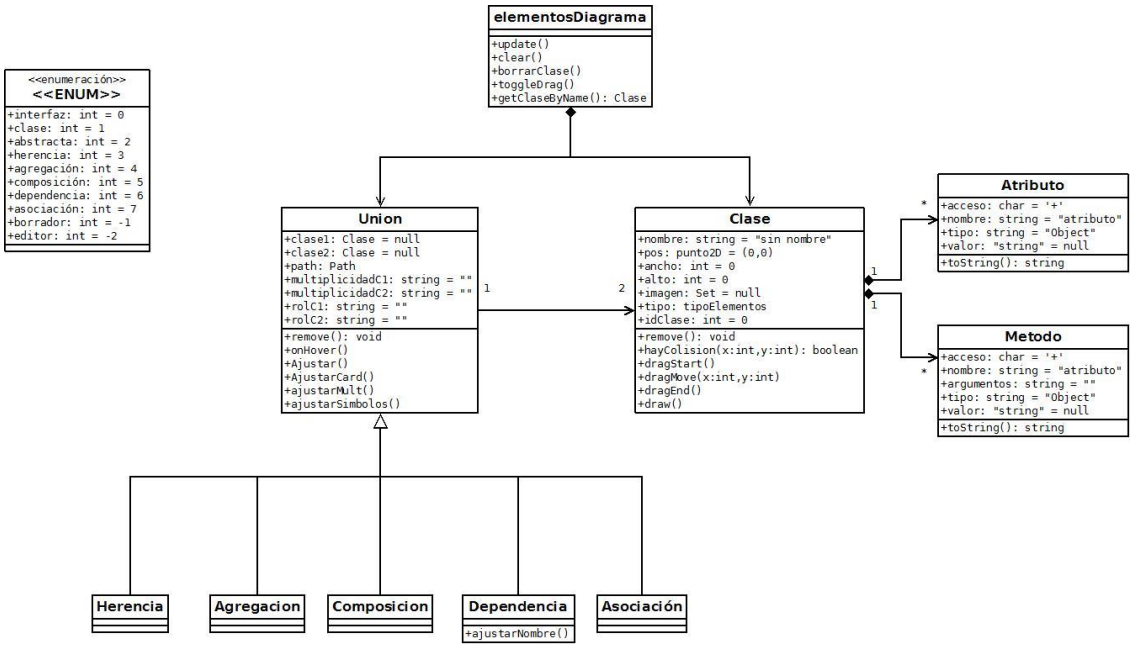


Ilustración 13: Diagrama de clases del editor de diagramas

Una de las funciones más importantes para el editor de diagramas es el método update(), que actualiza toda las representaciones gráficas de todos los elementos del diagrama. Esta función

es llamada, por ejemplo, cada vez que el usuario arrastra una clase para cambiar su posición, ya que las relaciones que pueda tener la clase tienen que ajustarse a la nueva posición. Lo que hace exactamente es recorrer los arrays de los elementos existentes y actualizar cada clase individualmente.

Ahora veamos cómo funciona el guardado de diagramas. Cuando un usuario guarda un diagrama se realizan dos peticiones Ajax. La primera comprueba si el nombre de un diagrama está disponible y la segunda, si no existe otro diagrama con el nombre o si el usuario confirma el sobrescrito, envía al servidor el diagrama convertido a formato JSON. El servidor lo guarda en un fichero en la carpeta de diagramas y el guardado se muestra como terminado al usuario si todo ha ido correctamente. Mientras dura este proceso el editor se bloquea y al usuario se le muestra una barra de progreso.

Cuando un usuario realiza el cargado de un diagrama, el proceso es el inverso al guardado. El servidor envía al cliente una tabla con los diagramas disponibles en la carpeta de diagramas guardados y, cuando el usuario selecciona uno de éstos, envía el diagrama en formato JSON, que el cliente recibe y recorre, instanciando cada uno de los datos del objeto recibido. Al igual que con el guardado, al usuario se le bloquea el editor y se le muestra una barra de progreso. Cuando todo termina, se informa con un mensaje sobre el diagrama cargado.

Cuando un usuario carga un diagrama dispone de una opción para mantener el contenido anterior del canvas. Es decir, si un usuario elige esta opción se cargaría el diagrama sobre los elementos existentes, de forma que si lo desea el profesor puede combinar diferentes diagramas sin necesidad de tener que volver a crear todos los elementos.

Toda la representación de elementos en el canvas donde se dibuja el diagrama de clases se hace con Raphael.js. Entre sus utilidades Raphael.js nos ofrece lo necesario para representar de forma sencilla elementos como polígonos, rectángulos, líneas y textos en un canvas HTML.

Comencemos viendo el proceso de creación de clases en el canvas. En primer lugar se dibujan los textos (nombre, métodos y atributos) y las líneas que separan nombre, atributos y métodos. Estos elementos definen las dimensiones de la nueva clase. Una vez todos se han representado, se toma el ancho del texto más largo y la suma de las alturas de todos los textos y líneas dibujados y se añade un rectángulo que tendrá la altura de la suma de las alturas de todos estos elementos y el ancho del texto o línea más largo. Una vez se han representado todos los elementos, se añaden a un set de Raphael.js. Un set no es más que una agrupación de elementos que hace que los cambios realizados a uno afecten a todos. Por último y ya generado el set, se guarda en la clase como imagen, terminando el proceso de representación de una clase en el canvas.

Se decidió que las relaciones seguirían una ruta en zigzag que se ajustara a la posición de sus clases. Esto hace que tengan un proceso de representación mucho más complejo, ya que tienen que calcular la ruta que van a seguir en tres partes, donde los puntos de cambio de dirección de la clase son relativos a la posición de las clases.

Para el cálculo de los puntos que componen una unión se ha utilizado el siguiente sistema: Primero se hallan los 4 posibles puntos que podría tener cada extremo de la relación en los lados de su clase; Tras esto, se seleccionan los puntos que guardan menor distancia euclidiana entre ellos. Por último, dependiendo de la combinación de lados de las clases a la que pertenezcan estos puntos seleccionados se establece la ruta que seguirá la unión.

Para las relaciones que tienen algún elemento extra, como flechas (herencias), rombos (por ejemplo las composiciones) y texto (multiplicidad, rol, etc), el añadido de estos elementos funciona de la misma forma, ya que tiene que situarse en el punto de unión de la clase con el extremo de la unión.

4.3 Modulo del sistema de gestión

4.3.1 Diseño del módulo del sistema de gestión

El sistema de gestión permitirá a los profesores, una vez hayan accedido con su usuario y contraseña, realizar un seguimiento de los alumnos, modificar elementos del juego y gestionar diagramas. No se mostrará un diagrama de navegación entre páginas porque en el sistema de gestión cualquier página puede ser accedida desde otra a través de un menú de navegación situado a la izquierda de la página. Este menú se puede ocultar para no perder siempre ese porcentaje de pantalla (ya que en pantallas como la del editor de diagramas es importante disponer del máximo de pantalla posible).

La pantalla principal, a la que se accede nada más entrar al sistema, contiene un pequeño resumen de todas las pantallas del sistema: El número de diagramas que hay, la cantidad de alumnos y administradores, la última conexión al sistema, la máxima puntuación de un alumno y el número de generadores disponibles.

A continuación se explican en detalle cada una de las páginas de este sistema, aunque no se hablará del editor de diagramas, ya que se ha explicado con todo detalle en la sección anterior.

4.3.1.1 Gestión de usuarios

Los usuarios se pueden encontrar en el sistema agrupados en dos tablas, una para los administradores (profesores, acceden al sistema de gestión) y otra para los alumnos (alumnos, acceden al juego). En cada fila se puede encontrar los datos de un usuario y al final una celda que permite la edición de un usuario y otra el borrado del mismo. Ambas tablas permiten realizar búsquedas, ordenar por las diferentes columnas y mostrar una cantidad variable de datos.

Ilustración 14: Pantalla de usuarios del sistema de gestión

Situados en la parte inferior de cada tabla, hay sendos botones que permiten crear un nuevo usuario. Con la edición de usuarios (tanto de profesores como de alumnos) se puede cambiar cualquier dato de la tabla salvo la ID del usuario.

4.3.1.2 Gestión de elementos del juego

Esta pantalla es muy similar a la anterior, salvo que en vez de usuarios y administradores se tienen modificadores (modificadores constantes) y bonificadores (modificadores temporales). Las tablas están implementadas del mismo modo, así que comparten todas las funcionalidades con las descritas en la anterior pantalla.

Ilustración 15: Pantalla de gestión de elementos de juego.

Para crear los elementos de juego se dispone de los siguientes Modal Box, donde se pueden definir cada uno de sus parámetros.

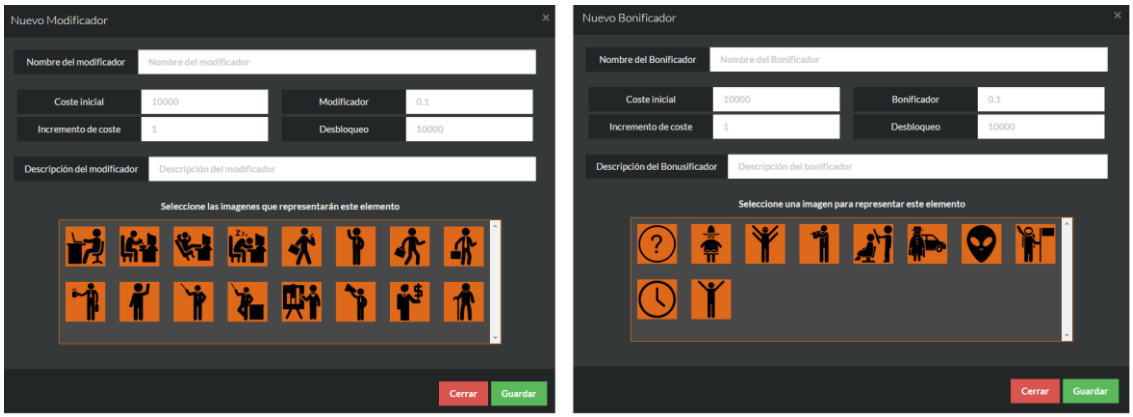


Ilustración 16: Modal Box de creación de modificadores

Las imágenes que se muestran como opciones para representar los elementos deben ser subidas manualmente al servidor.

4.3.1.3 Estado del juego

Aquí es donde el profesor puede hacer el seguimiento de los alumnos. En esta página se muestra relacionado cada alumno con su empresa del juego, mostrando valores de interés, como su porcentaje de éxito, los diagramas que ha resuelto con éxito, los que ha intentado, el

valor de su empresa, etc. También se añade el correo del usuario por si se quiere contactar con él y al final de cada fila un botón para borrar la empresa.

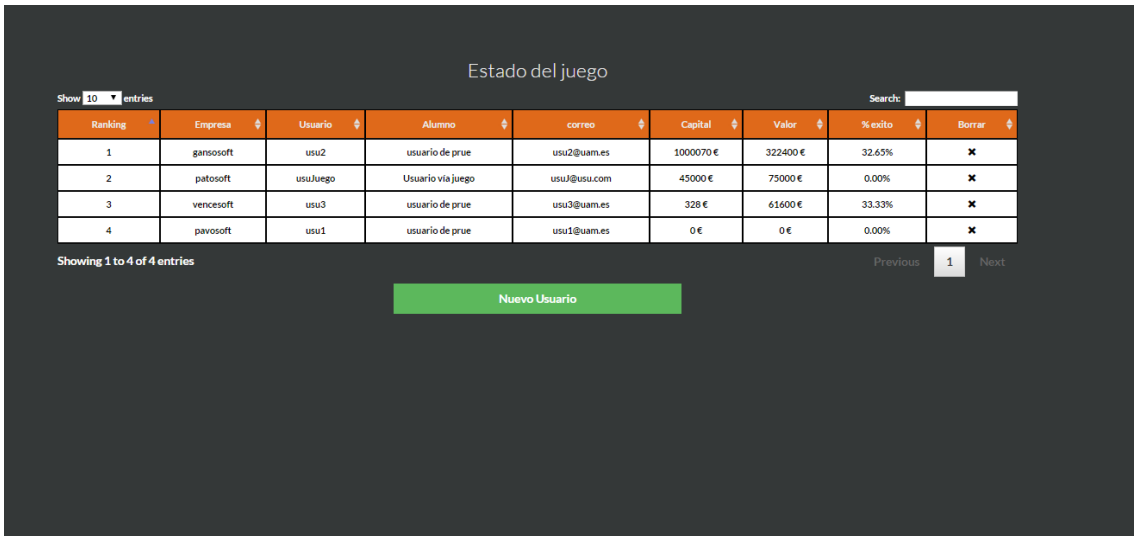


Ilustración 17: tabla con la información del seguimiento de los alumnos

4.3.2 Implementación del módulo de gestión

Todas las tablas están creadas usando el plugin DataTables Jquery, que les añade la funcionalidad de ordenar, buscar y paginar.

Las tablas que se rellenan desde PHP con información de la base de datos, como las tablas de usuarios y la tabla de estado de juego, se rellenan recorriendo todas las filas de sus tablas correspondientes en la base de datos.

Por otro lado las tablas que se rellenan con la información obtenida de ficheros JSON, como las tablas de modificadores, recorren todos los ficheros de las carpetas donde se encuentran estos elementos, leyendo uno a uno y añadiendo su correspondiente fila a la tabla.

4.4 Base de datos

Se ha escogido una base de datos relacional para el almacenamiento de la información de los usuarios, administradores y estado del juego. Todos los accesos a la base de datos se hacen desde PHP usando PDO, ya que permite conectarse a cualquier base de datos relacional de forma segura [19].

Para cada función acceso a la base de datos se ha implementado un servicio que puede ser llamado con AJAX y que responde con un objeto JSON si se trata de un select, vacío si es un insert, delete o update o un error si es que se ha producido algún problema. Las funciones que necesitan obtener datos para llevar a cabo su operación también reciben objetos JSON. Los parámetros de conexión a la base de datos pueden cambiarse desde un fichero PHP donde se encuentran una serie de variables que usa la aplicación para crear las conexiones.

La aplicación tiene en sus directorios un fichero dump.sql donde vienen todas las instrucciones para crear todas las tablas necesarias y unos cuantos datos para realizar pruebas.

A continuación se muestra un esquema de la base de datos y se da una breve explicación del uso de cada tabla.

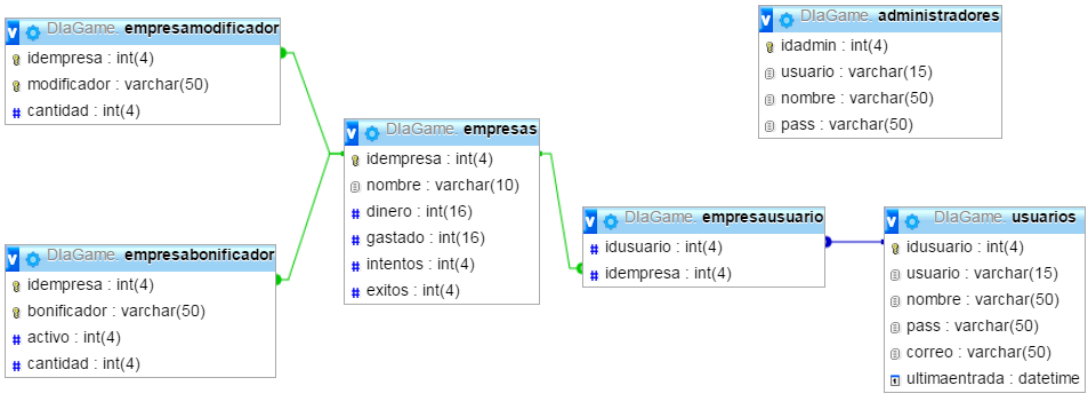


Ilustración 18: Esquema de la base de datos

- **Administradores**: Guarda la información de los usuarios de los profesores que será utilizada para acceder al sistema de gestión.

- **Usuarios**: Guarda la información de los usuarios de los alumnos que serán utilizados para acceder al sistema y para llevar a cabo el seguimiento por los profesores.

- **Empresas**: Guarda el estado de la partida de un usuario. Es decir, guarda toda la información de la empresa de cada usuario.

- **Empresausuario**: Guarda una relación de qué empresa de la tabla empresas pertenece a que usuario de la tabla usuarios.

- **Empresamodificador**: Guarda una relación entre una empresa y los modificadores constantes que posee adquirido.

- **Empresabonificador**: Guarda una relación entre una empresa y los modificadores temporales que la empresa tenga activos.

4.5 Elementos del juego

Con elementos de juego nos referimos a los diagramas de clases generados por los profesores y a los modificadores temporales y constantes. Todos estos elementos se guardan individualmente en ficheros JSON en diferentes carpetas.

El motivo de que estos elementos no se almacenen es que se especificó en los requisitos del sistema se puede exportar los elementos del juego de un sistema otro sencillamente. Por este motivo los elementos fueron implementados en JSON.

De esta forma, basta con sustituir las carpetas que contienen estos elementos del sistema que queramos exportar al nuestro. SI en la exportación desaparece algún fichero de modificador del sistema, no provocaría ningún error ni haría desaparecer estos elementos de la base de datos, simplemente los ignoraría, por si en algún momento se necesitará la información.

5. Pruebas

Cada vez que se terminaba un módulo de Diagames se llevaban a cabo una serie de pruebas para localizar errores y comprobar la correcta funcionalidad del sistema. En esta sección se irán explicando las pruebas realizadas por módulos.

5.1 Pruebas del editor de diagramas

En el módulo de edición de diagramas, es importante que la funcionalidad del editor sea correcta, pero también es importante que se compruebe toda la parte de guardado y carga de diagramas, ya que de esta parte depende el correcto funcionamiento de los problemas en el juego. Por esto, se realizaron las siguientes pruebas:

Objetivo a probar	Prueba	Resultados
Clases	Crear clases de tamaño variable, con diferentes números de atributos y de métodos	La clase se adapta correctamente
	Crear una clase sin métodos, sin nombre y sin atributos	Se genera una clase con el nombre “sin nombre”
	Arrastrar una clase y moverla, también probar a sobreponerla a otros elementos	Se mueve al lugar deseado y al colocarla sobre otras clases queda por encima. Si se coloca sobre relaciones queda debajo de estas
	Comprobar todos los tipos posibles de clase	Se genera el tipo especificado al crear la clase
Relaciones	Unir dos clases con una relación y mover estas clases. Hacer esto para todas las relaciones posibles	La relación se genera correctamente, al mover las clases la relación se adapta a sus posiciones
	Probar todas las estrategias de unión entre clases, tanto por defecto como manualmente al crear la relación	El punto de unión representado en el diagrama se corresponde a lo especificado
Borrador	Borrar una clase y una clase que posea relaciones con otra clase	La clase desaparece del diagrama. Las relaciones son borradas junto con la clase.
	Borrar una relación	La relación desaparece del diagrama
Editor	Editar todos los datos de una clase	Los valores se cambian correctamente, las relaciones se adaptan al nuevo tamaño de la clase
	Editar todos los datos de una relación. Hacer esto con todas las relaciones posibles	Para todos los tipos de relación se cambian los valores
Guardado	Guardar un diagrama cualquiera con elementos y alguna dimensión especifica	Se crea un fichero JSON en la carpeta saves con todos estos datos estructurados correctamente y con el nombre indicado
	Guardar un diagrama vacío	Se genera el fichero JSON igualmente
	Guardar un diagrama sin nombre	El sistema no permite el guardado

	Guardar un diagrama con un nombre que ya tenga un diagrama existente	El sistema pide confirmación para sobrescribir al usuario. Si se confirma genera el fichero JSON correctamente
Cargado	Cargar un diagrama cualquiera con o sin elementos	En todos los casos el diagrama se presenta en el editor sin modificaciones desde que se guardó
	Borrar parte de un fichero JSON de un diagrama y luego intentar cargarlo	El sistema advierte de que el diagrama no pudo ser cargado
	Crear un diagrama. Luego cargar otro, borrando los elementos que ya existían	Aparece el diagrama cargado y desaparece el previo
	Crear un diagrama. Luego crear uno nuevo, marcando la opción de combinar diagramas	Aparece el diagrama cargado junto con el previo
Borrado	Borrar un diagrama cualquiera	El diagrama se borra correctamente y desaparece del sistema de gestión y del juego
Guardado, cargado y borrado	Intentar invocar los métodos de guardado y cargado manualmente desde fuera del sistema	El sistema responde con un error ya que no tiene sesión
Nuevo diagrama	Crear un diagrama y luego crear uno nuevo que borre lo anterior	Los elementos que había en el canvas desaparecen, el canvas se queda en blanco
Modo tutorial	Alternar entre el modo tutorial ON y OFF	Todos los mensajes de ayuda se muestran en tutorial ON pero no en tutorial OFF

Tabla 1: Lista de pruebas realizadas en el módulo de edición de diagramas

5.2 Pruebas del módulo de juego

En el módulo de juego, además de comprobar el correcto funcionamiento, lo más importante será probar que la corrección siempre da una evaluación correcta, que los generadores funcionan correctamente y que el usuario no puede modificar maliciosamente los datos de una partida. Por esto se han realizado las siguientes pruebas

Objetivo a probar	Prueba	Resultados
Pantalla principal	La pantalla muestra los datos correspondientes al usuario al cargar si ya tiene empresa, si no pide nombre para una empresa	Pide nombre de empresa si no existe, luego los datos cargados concuerdan con la BD y los ficheros JSON
	Adquirir modificadores temporales y constantes	Los datos de la empresa se actualizan, así como el canvas y el precio de los elementos
	Intentar adquirir modificadores temporales y constantes sin dinero suficiente	El sistema no permite la compra y muestra la animación de acción denegada
	Intentar modificar con JavaScript los	Los cambios se realizan en el cliente,

	valores de la empresa	pero a la hora de guardar el sistema no realiza cambios en la BD, por lo que a la próxima entrada del usuario, los datos no están modificados
Ranking	Cambiar desde la BD los valores de las empresas	Los cambios que afectan al ranking se ven representados en el Modal Box de ranking
Generadores de problemas	Comprobar que todos los diagramas y generadores son cogidos al azar al menos alguna vez y que generan de forma correcta	Todos los generadores son llamados, los cambios en los ficheros JSON de problemas concuerdan con el generador usado y con el diagrama alterado. La descripción del generador y el enunciado del diagrama concuerdan
	Comprobar que se genera correctamente la solución con el problema	Junto con el fichero JSON alterado aparece el diagrama sin alterar en otro fichero
	Quitar todos los ficheros PHP de los generadores de diagramas.	No se produce un error, pero se genera un fichero sin alterar como problema
	Quitar todos los ficheros JSON de diagramas	No se produce un error, pero se genera un diagrama vacío
Resolución del problema	Dar por terminada la resolución de un problema	Se genera un fichero JSON con el diagrama dado como respuesta por el alumno
	Salir del editor de diagramas antes de dar por terminada la resolución. Intentar recargar la página	Se le muestra una advertencia al usuario y si confirma, al salir se le da por fallido el diagrama. Esto funciona aunque se cierre la página
Corrector	Comprobar que todos los errores cuadran con el diagrama dado como respuesta y que la valoración del diagrama es correcta	En todas las pruebas realizadas cuadra
	Obtener un resultado fallido en el problema	Al ser el error mayor de 5, se muestra el mensaje de fallido y no se obtienen beneficios
	Obtener un resultado de correcto en el problema	Al ser el error menor de 5, se muestra un mensaje de correcto y se obtienen beneficios, con los correspondientes modificadores
	Recargar la página o llamarla para intentar aumentar nuestros beneficios con trampas	Muestra un mensaje de página no disponible

Tabla 2: Lista de pruebas realizadas en el módulo de juego

5.3 Pruebas del módulo de gestión

En el módulo de gestión lo más importante es que el seguimiento se muestre correctamente y que se puedan manipular los usuarios sin posibilidad de error, al igual que los elementos de juego. Por ello las pruebas relazadas fueron:

Objetivo a probar	Prueba	Resultados
Gestión de usuarios	Comprobar que los datos mostrados de la tabla de usuarios y de administradores son correctos	Los datos concuerdan con la BD
	Crear un usuario y un administrador	Los usuarios figuran correctamente en las tablas y en la BD
	Editar los datos de un usuario o administrador.	Los datos aparecen cambiados correctamente en la BD y en las tablas
	Borrar un usuario y un administrador	Los usuarios desaparecen tanto de BD como de las tablas
	Borrar el usuario con el que se está conectado	Al recargar la página nos envía a la pantalla de acceso, pues el usuario ya no existe
Seguimiento	Los datos mostrados en la tabla de seguimiento son correctos	Coinciden con los datos de la BD.
Elementos de juego	Comprobar que los datos mostrados de la tabla de modificadoras y de bonificadores del juego son correctos	Los datos concuerdan con los datos de los ficheros JSON
	Crear un modificador de cada clase	Se añaden a la tabla y se generan los ficheros JSON correctamente
Pantalla principal	Comprobar que los datos de resumen concuerdan con los datos reales	Los datos coinciden con lo mostrado en las otras pantallas

Tabla 3: Lista de pruebas realizadas en el módulo de gestión

5.4 Pruebas generales

Es importante para todo el sistema que no se pueda alterar componentes de la plataforma sin haber accedido previamente al sistema y que los usuarios puedan registrarse y acceder sin problemas al juego. Las pruebas realizadas han sido:

Objetivo a probar	Prueba	Resultados
Acceso al sistema	Comprobar el acceso al sistema de juego y de gestión	Los administradores pueden acceder únicamente al sistema de gestión y los usuarios al de juego
	Intentar acceder a cualquier página sin haber accedido con credenciales	Las páginas y servicios rechazan el acceso al no tener en sesión una idusuario o idadministrador.
Registro de alumnos	Registrar alumnos desde la pantalla de acceso del juego	Se rellena un formulario para crear un usuario. En este formulario se comprueba que los datos sean válidos. El nuevo usuario aparece en la BD y tiene acceso al juego

Tabla 4: Lista de pruebas generales realizadas

6. Conclusiones

6.1 Conclusiones

Este ha sido un proyecto personalmente motivador ya que siempre había sentido curiosidad por la gamificación. Gracias a este trabajo he podido entrar de lleno en este tema. Está claro que la gamificación tiene un gran potencial y que en un futuro su uso va a ser muy amplio.

El trabajo realizado ha permitido cubrir todos los objetivos planteados al iniciar este Trabajo Fin de Grado: Se ha desarrollado una plataforma completa, que permite a los alumnos aprender diseño orientado a objetos con técnicas de gamificación, a través de un juego basado en web y que además permite a los profesores realizar un seguimiento de los alumnos en un juego que pueden modificar a su gusto.

El sistema no se ha puesto en producción por falta de tiempo, pero podría resultar útil en asignaturas como Análisis y Diseño de Software y Proyecto de Análisis y Diseño de Software. Además de esta forma se podría comprobar si las técnicas de gamificación aplicadas realmente motivan a los alumnos a practicar más que con la típica lista de ejercicios.

Creo que he adquirido gran cantidad de conocimientos no solo sobre la gamificación, sino sobre todas las tecnologías usadas, sobre todo con JavaScript y PHP que apenas se ve a lo largo de la carrera, pero que han resultado ser unos lenguajes muy interesante y creo que me serán útiles en un futuro. Además de esto, he tenido oportunidad de tener contacto con librerías hasta ahora desconocidas para mí (como son todas las que se han usado, salvo JQuery) y que creo que podrían hacer aún más valiosos los conocimientos adquiridos.

6.2 Posibles ampliaciones

Durante el desarrollo del trabajo fueron surgiendo una serie de ideas que no se contemplaban inicialmente en los requisitos, pero que mejorarían notablemente el sistema:

- **Reloj**: Una cuenta atrás que limite el tiempo que puede dedicar un usuario a resolver un diagrama. Este tiempo iría en proporción inversa al porcentaje de éxito y al valor de la empresa del usuario y añadiría beneficios a la resolución del diagrama en base al tiempo tardado. También se podría añadir un modificador temporal que otorgase tiempo extra al usuario.

- **Log de sucesos**: Añadir otra pestaña al sistema de gestión que muestre a los profesores una tabla con eventos relevantes del sistema. Aquí se mostraría, por ejemplo, si se ha detectado que un usuario ha hecho trampas, errores o excepciones que se produzcan en el sistema y eventos menos relevantes, como que diagramas con que generadores resuelve cada usuario.

- **Mensajes**: En vez de mostrar el correo de usuario en la pantalla de seguimiento, estaría bien que el profesor pudiera contactar con el usuario directamente a través del sistema.

- **Añadir opciones a los usuarios**: Actualmente el sistema no permite a los usuarios cambiar sus datos o contraseña. También se debería añadir a las páginas de acceso una opción para recuperar contraseñas olvidadas.

- **Sistema de pistas**: Ofrecer a los alumnos la posibilidad de obtener pistas para resolver el diagrama. Por cada pista solicitada se penalizaría la ganancia total del problema.

- **Gráficos**: Añadir a la página de inicio del sistema de gestión unos graficas que permitan a los profesores hacerse una idea rápida del estado del sistema. Por ejemplo los usuarios que entran por día, los valores de las empresas, los porcentajes de éxito medios a través del tiempo, etc. Esto podría ser implementado sencillamente con Morris.js [17].

- **Mejorar el sistema de BD**: Sería interesante que en vez de tener que usar manualmente el fichero dump.sql para generar en la base de datos las tablas necesarias para la aplicación, hubiera alguna opción que permitiera generar estas tablas automáticamente y otras opciones relacionadas, como por ejemplo limpiar todos los datos existentes.

- **Descargar diagramas**: Añadir una opción en el Modal Box de guardado de diagramas que permitiera bajarse el fichero JSON que corresponde al diagrama. Esto sería útil, por ejemplo, si el servidor no está disponible al guardar por problemas técnicos. De esta forma no se perdería todo el trabajo realizado.

- **Subida de imágenes para modificadores**: Las imágenes se tienen que subir manualmente a una carpeta del servidor. Habría que dar una opción del sistema de gestión que permita añadir imágenes a los profesores sin necesidad de esto.

7. Referencias

7.1 Libros de interés

[1] Gamification: A Simple Introduction.
(2013) Andrzej Marczewski.

[2] Gamification in Education and Business.
(2012) Karl M. Kapp.

7.2 Enlaces

[3] Animate.css,
https://daneden.github.io/animate.css/

[4] BBVA game,
https://www.bbva.es/particulares/subhome/bbva-game-juega/index.jsp

[5] Bootstrap,
http://getbootstrap.com/

[6] ClassDojo,
https://www.classdojo.com/

[7] Cookie Clicker,
http://orteil.dashnet.org/cookieclicker/

[8] Correos,
http://www.wonnova.com/casos-de-exito

[9] Coursera,
https://www.coursera.org/

[10] DataTables,
https://www.datatables.net/

[11] Duolingo,
https://play.google.com/store/apps/details?id=com.duolingo&hl=es

[12] Ejemplos de gamificación en empresas,
http://www.elconfidencial.com/empresas/2014-04-27/gamificacion-o-como-lograr-que-los-empleados-hagan-un-trabajo-extra-gratis_121168/

[13] Intermatia,
https://www.intermatia.com/home.php

[14] Inversiones en gamifiación,
http://www.gamesindustry.biz/articles/2012-05-21-gamification-market-to-reach-USD2-8-billion-in-2016

[15] JQuery,
https://jquery.com/

[16] Jquery-MD5,
https://github.com/placemarker/jQuery-MD5

[17] Morris.js
http://morrisjs.github.io/morris.js/

[18] Pain Squad Mobile App,
http://www.campaignpage.ca/sickkidsapp/

[19] PHP PDO
http://php.net/manual/es/book.pdo.php

[20] Sobre el BBVA game,
http://omniumgames.com/bbva-game-el-mayor-caso-de-exito-de-la-gamificacion-en-espana/

[21] Sobre Stack Overflow,
http://library.fora.tv/2012/06/21/Stack_Overflow__Stack_Exchange_Programming_Programmers

[22] Steam,
http://blog.inerciadigital.com/2013/10/08/gamificacion-un-vistazo-a-steam/

[23] Tiempo medio gastado en videojuegos en EEUU,
http://bgr.com/2014/05/14/time-spent-playing-video-games/

[24] Tooltipster,
http://iamceege.github.io/tooltipster/

[25] World Peace Game,
http://worldpeacegame.org/

[26] Zombies, Run!,
https://zombiesrungame.com/

8. Anexos

8.1 Anexo 1: Manual de programador

8.1.1 Añadiendo Nuevos generadores de problemas

Diseñar un nuevo generador es algo relativamente sencillo. Para ello hay que crear un nuevo fichero PHP con el nombre que tendrá el generador y guardarlo en la carpeta del proyecto situada en **Diagame/juego/php/generadores/.** Esta carpeta será recorrida por el sistema a la hora de escoger un generador para formar un problema.

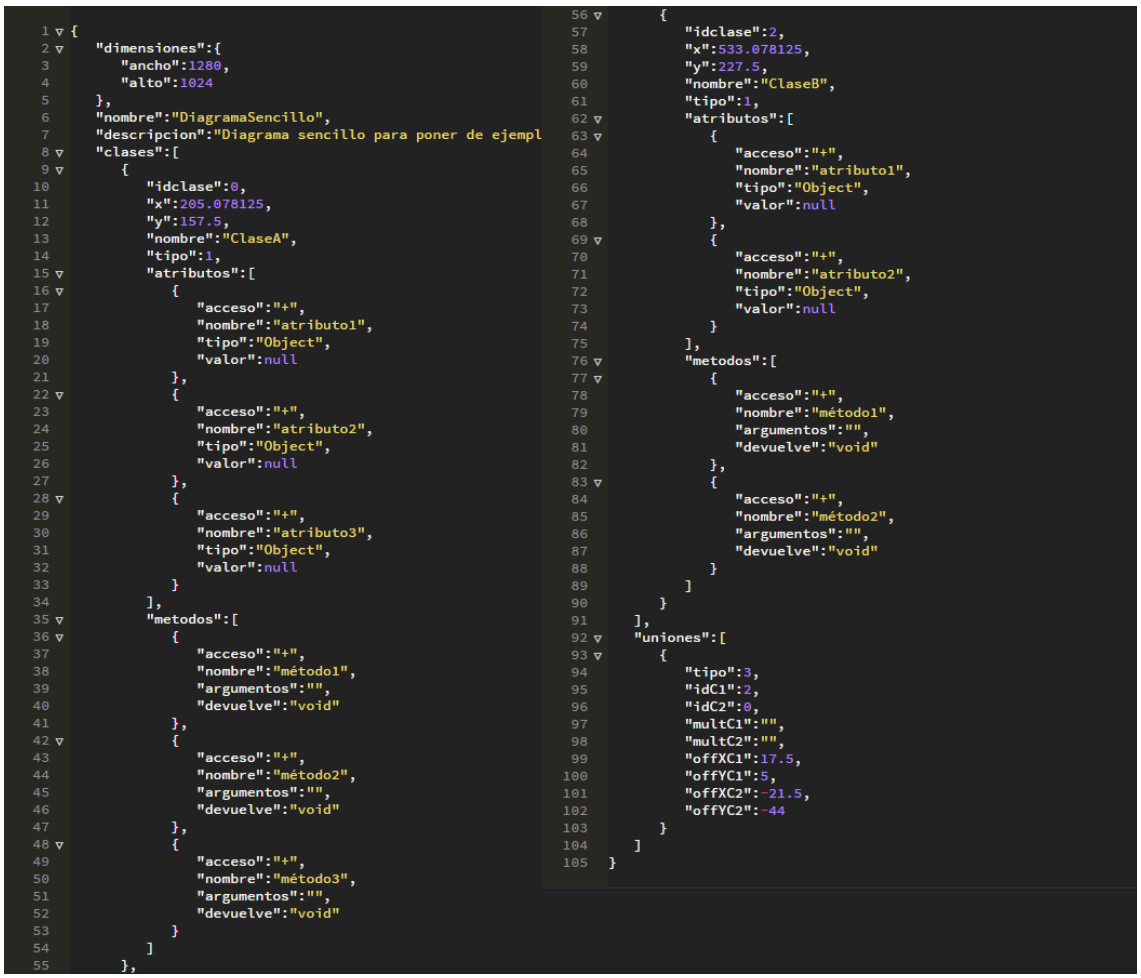
Una vez hecho esto, podemos comenzar a implementar el generador. Lo primero de todo hay que actualizar la variable $descripcion con un String que contenga información útil para el alumno sobre qué tipo de alteraciones se han hecho en el diagrama. Tras esto podemos empezar a alterar el diagrama como deseemos, al cual se puede acceder como un objeto JSON llamado $obj. Para más información sobre la estructura de los diagramas consultar el anexo Estructura de los diagramas.

```php
<?php
    // Los generadores tienen que usar la variable $obj
    // que contiene los datos de el daigrama en JSON.
    // También tienen que sobreescribir la variable $descripcion
    // con una explicación de lo que hace el generador para ayudar
    // a los alumnos a resolver los problemas generados.

    $descripcion = "Corrige el acceso si es necesario de los métodos y atributos de las clases del diagrama.";

    // Recorrer todas las clases
    foreach($obj->clases as $clase){

        $accesos = ['#','+','-'];

        // Recorrer todos sus atributos y cambiar al azar la visibilidad de éstos
        foreach($clase->atributos as $atributo){
                $rand = rand(0,2);
                $atributo->acceso = $accesos[$rand];
        }

        // Recorrer todos sus métodos y cambiar al azar la visibilidad de éstos
        foreach($clase->metodos as $metodo){
                $rand = rand(0,2);
                $metodo->acceso = $accesos[$rand];
        }

    }


?>
```

Ilustración 19: Código de ejemplo de un generador

8.1.2 Estructura de los diagramas

Los diagramas son objetos JSON que se guardan en la carpeta del proyecto **Diagame/gestión/saves/**. La mejor forma de entender esta estructura es con un ejemplo:




```
{
   "dimensiones":{
      "ancho":1280,
      "alto":1024
   },
   "nombre":"DiagramaSencillo",
   "descripcion":"Diagrama sencillo para poner de ejempl
   "clases":[
      {
         "idclase":0,
         "x":205.078125,
         "y":157.5,
         "nombre":"ClaseA",
         "tipo":1,
         "atributos":[
            {
               "acceso":"+",
               "nombre":"atributo1",
               "tipo":"Object",
               "valor":null
            },
            {
               "acceso":"+",
               "nombre":"atributo2",
               "tipo":"Object",
               "valor":null
            },
            {
               "acceso":"+",
               "nombre":"atributo3",
               "tipo":"Object",
               "valor":null
            }
         ],
         "metodos":[
            {
               "acceso":"+",
               "nombre":"método1",
               "argumentos":"",
               "devuelve":"void"
            },
            {
               "acceso":"+",
               "nombre":"método2",
               "argumentos":"",
               "devuelve":"void"
            },
            {
               "acceso":"+",
               "nombre":"método3",
               "argumentos":"",
               "devuelve":"void"
            }
         ]
      },
      {
         "idclase":2,
         "x":533.078125,
         "y":227.5,
         "nombre":"ClaseB",
         "tipo":1,
         "atributos":[
            {
               "acceso":"+",
               "nombre":"atributo1",
               "tipo":"Object",
               "valor":null
            },
            {
               "acceso":"+",
               "nombre":"atributo2",
               "tipo":"Object",
               "valor":null
            }
         ],
         "metodos":[
            {
               "acceso":"+",
               "nombre":"método1",
               "argumentos":"",
               "devuelve":"void"
            },
            {
               "acceso":"+",
               "nombre":"método2",
               "argumentos":"",
               "devuelve":"void"
            }
         ]
      }
   ],
   "uniones":[
      {
         "tipo":3,
         "idC1":2,
         "idC2":0,
         "multC1":"",
         "multC2":"",
         "offXC1":17.5,
         "offYC1":5,
         "offXC2":-21.5,
         "offYC2":-44
      }
   ]
}
```


Ilustración 20: Estructura JSON de un diagrama de clases

Como se puede ver, el diagrama está organizado de forma que en el primer nivel del objeto JSON encontramos los datos básicos del diagrama: dimensiones (ancho y alto), nombre y descripción. Tras esto tenemos un array de clases y un array de relaciones llamado uniones.

El array de clases está formado por objetos que contienen la información necesaria para instanciar y representar una clase: el identificador de la clase (número entero que se asigna al ser creada en el diagrama), las coordenadas en el canvas, el nombre y tipo de clase (interfaz: 0, clase: 1, abstracta: 2), sus métodos y sus atributos. A su vez, métodos y atributos se guardan en arrays con la información individual de cada uno: acceso, nombre, tipo y valor por defecto para los atributos y acceso, nombre, argumentos y tipo devuelto para métodos.

El array de uniones guarda un objeto por relación presente en el diagrama. La información que compone una unión consta del tipo de relación (herencia: 3, agregación: 4, composición: 5, dependencia: 6, asociación: 7), de los identificadores de las clases en las que participa, de la multiplicidad de cada extremo de la relación y de las coordenadas (relativas al punto medio del lateral de una relación) de conexión para cada relación. Aunque en el ejemplo no se encuentren presente porque es una relación de herencia, también se almacena el rol de cada extremo y el nombre, en el caso de las dependencias.

8.1.3 Estructura de los elementos del juego

Las estructuras para un elemento que aporta un modificador permanente al juego y otro que aporte un modificador temporal son muy similares. Aun así, con el fin de aclarar cualquier duda, se mostrará un ejemplo de cada.

La primera diferencia entre estos objetos es que se encuentran localizados en carpetas del proyecto diferentes. Se pueden encontrar los modificadores permanentes en la carpeta **Diagame/elementosjuego/modificadores/** y los modificadores temporales en la carpeta **Diagame/elementosjuego/bonus/.**

Empecemos analizando la estructura de los modificadores permanentes. Cada fichero JSON contiene el nombre del modificador, la descripción del mismo, su coste inicial, el modificador que aporta a los beneficios por unidad, el incremento del precio por unidad comprada, la cantidad necesaria para desbloquear este modificador y un array con las rutas de las imágenes que serán usadas aleatoriamente para representar este elemento.

```
{
    "modificador": [{
        "nombre": "Becario",
        "descripcion": "Un trabajador barato pero un poco distraido.",
        "coste": "6000",
        "modificador": "0.05",
        "incremento": "0.2",
        "desbloqueo": "0",
        "imagen": ["..\/juego\/img\/modificadores\/lvl1\/2.png",
"..\/juego\/img\/modificadores\/lvl1\/1.png",
"..\/juego\/img\/modificadores\/lvl1\/0.png",
"..\/juego\/img\/modificadores\/lvl1\/3.png"]
    }]
}
```

Ilustración 21: Estructura JSON de un elemento de juego permanente

Como se puede ver en los empleos, las estructuras son muy similares, solo que en vez de modificador, se refiere al elemento y al modificador que aporta al ser usado como bonificador.

```
{
    "bonificador": [
        {
            "nombre": "Tacos para todos!",
            "descripcion": "Invita a tacos a tus empleados! empleados
contentos, empleados productivos! ayayay!",
            "coste": "15000",
            "bonificador": "1.0",
            "incremento": "1",
            "desbloqueo": "20000",
            "imagen": [
                "../juego/img/bonus/0.png"
            ]
        }
    ]
}
```

Ilustración 22: Estructura JSON de un elemento de juego temporal

8.2 Anexo 2: Vista del juego en dispositivos móviles

A continuación se muestran una serie de pantallazos hechos en un dispositivo Android de la pantalla principal del juego para mostrar cómo se adapta la página a este tipo de resoluciones.

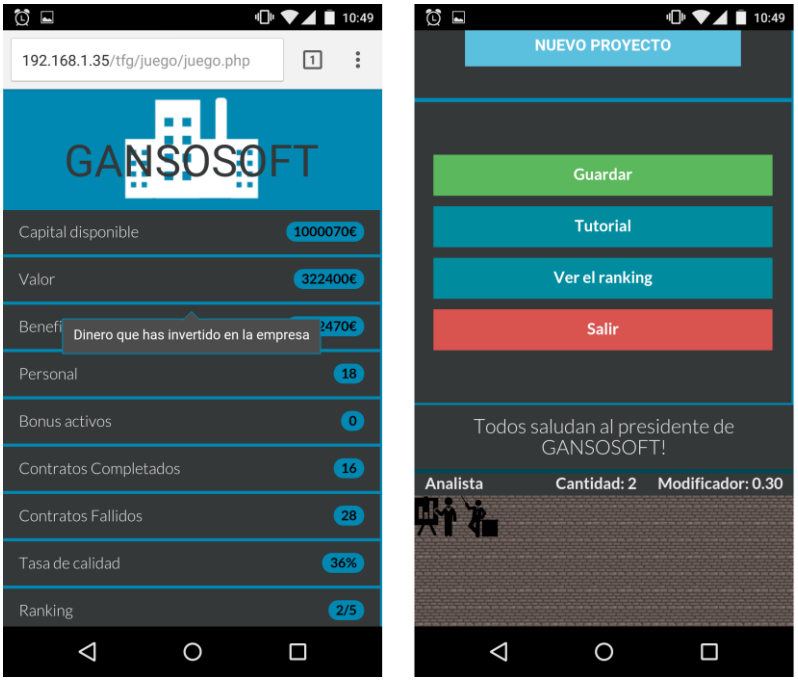


Ilustración 23: Pantalla de Juego en dispositivos móviles I

Se nos presenta como una página web alargada, donde se han situado lo que son las tres partes principales de la página principal una debajo de otra. Todas las interacciones han sido adaptadas a pantalla táctil.

Ilustración 24: Pantalla de Juego en dispositivos móviles II

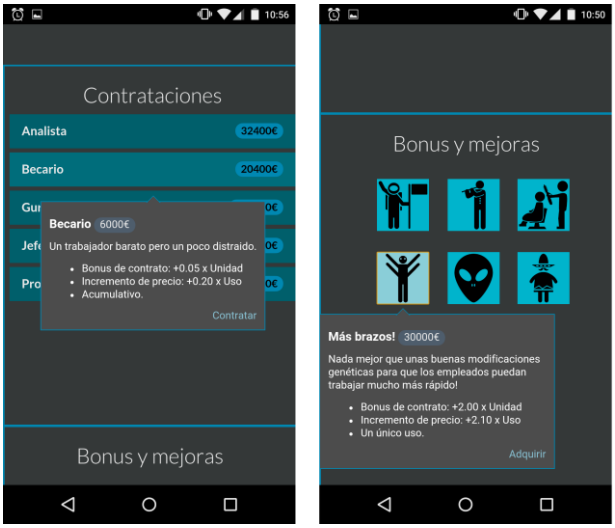


Ilustración 25: Pantalla de Juego en dispositivos móviles III

8.3 Anexo 3: Modal Box de elementos del editor de diagramas

En este apartado se muestran ejemplos de los Modal Box que se usan en el editor de diagramas para crear relaciones y clases.

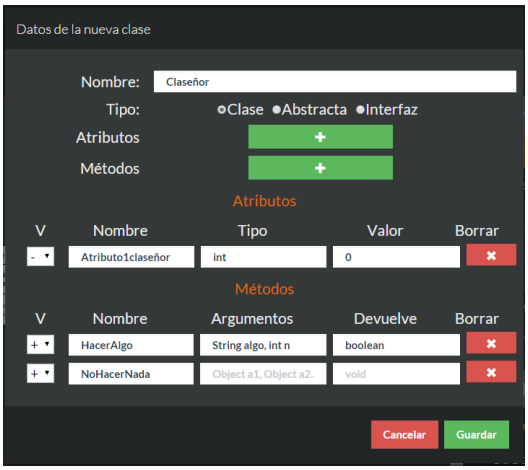


Ilustración 26: Ejemplo de Modal Box de creación de clase

Ilustración 27: Ejemplo de Modal Box de creación de relación